Commodore
COMPUTER CLUB
43
L. 4.000
La rivista degli utenti di sistemi Com
25 Giugno 1987 - Anno VI - N° 43 - Sped. Abb. Post. Gr. III/70 - CR - Distr. MePe
Cara Amiga, ti scrivo
GW-Basic con pochi spiccioli
Minicopiatore di programmi
Quando uno schermo non basta
Viaggio nella traccia 18
In anteprima il modem del Videotel
systems

Sommario

INSERTO

VIAGGIO NEL BASIC

RUBRICHE

Copertina: immagine da Linea intima 4/86, per gentile concessione di Kentelle.

Direttore: Alessandro de Simone - **Caporedattore:** Michele Maggi
Redazione/collaboratori: Paolo Agostini, Claudio Baiocchi, Carlo e Lorenzo Barazzetta, Simone Bettoia, Luigi Callegari, Sergio Camici, Sandro Certi, Umberto Colapicchioni, Maurizio Dell'Abate, Valerio Ferri, Giancarlo Mariani, Roberto Marigo, Clizio Merli, Marco Miotti, Roberto Morassi, Antonio Pastorelli, Carla Rampi, Marco Saetta, Fabio Sorgato, Danilo Toma, Giovanni Valli.
Segretaria di redazione: Maura Ceccaroli
Ufficio Grafico: Arturo Ciaglia, Gabriella Galbusera
Direzione, redazione, pubblicità: v.le Famagosta, 75 - 20142 Milano - Tel. 02/8467?48
Pubblicità: Milano: Leandro Nencioni (direttore vendite), Guido Agosti, Giorgio Ruffoni, Claudio Tidone - v.le Famagosta, 75 - 20142 Milano - Tel. 02/8467348
• Emilia Romagna: Spazio E - P.zza Roosevelt, 4 - 40123 Bologna - Tel. 051/236979
• Toscana, Marche, Umbria: Mercurio srl - via Rodari, 9 - San Giovanni Valdarno (Ar) - Tel. 055/947444
• Lazio, Campania: Spazio Nuovo - via P. Foscari, 70 - 00139 Roma - Tel. 06/8109679
Segreteria: Marina Vantini - **Abbonamenti:** Liliana Spina
Tariffe: prezzo per copia L. 4.000. Abbonamento annuo (11 fascicoli) L. 40.000. Estero: il doppio. Abbonamento cumulativo alle riviste Computer e Commodore Computer Club L. 80.000.
I versamenti vanno indirizzati a: Systems Editoriale Srl mediante assegno bancario o utilizzando il c/c postale n. 37952207
Composizione: Systems Editoriale Srl - **Fotolito:** Systems Editoriale Srl
Stampa: La Litografica Srl - Busto Arsizio (Va)
Registrazioni: Tribunale di Milano n. 370 del 2/10/82 - Direttore Responsabile: Michele Di Pisa
Sped. in abb. post. gr. III - Pubblicità inferiore al 70% - **Distrib.:** MePe - via G. Carcano, 32 - Milano

l'argomento del mese

Dalla fine all'inizio

Un periodo volge al termine ed uno nuovo inizia (e altre considerazioni)

Il periodo estivo, fino a settembre, è considerato di pausa, per qualsiasi attività.

Anche noi non sfuggiamo alla "regola" e ci sentiamo vicini agli studenti impegnati nello sprint finale, agli adulti che, durante la pausa di lavoro, scambiano idee con i colleghi sui prossimi luoghi di villeggiatura.

L'euforia delle elezioni è finita (?), lo scalpore suscitato dall'IBM per i suoi nuovi computer non fa più notizia (ne ha già fatta, e molta) ed i comunicati della Commodore, per ciò che ci riguarda, lasciano sperare in qualcosa di buono.

Altri avvenimenti hanno contribuito, recentemente, a destare la nostra attenzione; alcuni di questi sono legati al mondo del computer, altri un po' meno, ma tutti sono vicini, comunque, al campo del "progresso".

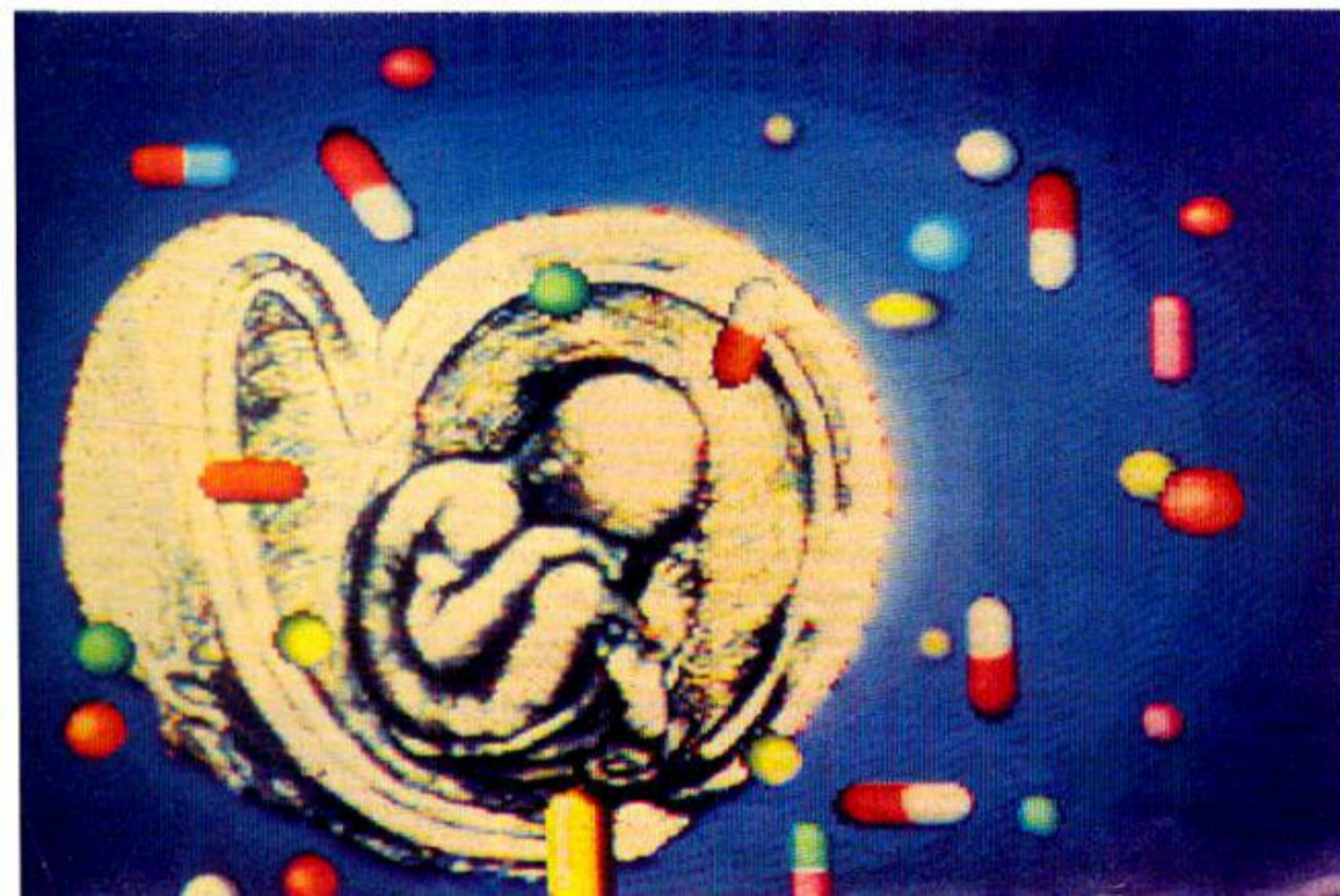
Più preoccupazione, invece, destano le recenti notizie sulle manipolazioni genetiche che porterebbero alla "creazione" di una razza subumana al servizio, è inutile dirlo, dei suoi "creatori".

Come dire che il razzismo, cacciato dalla porta, rientra indisturbato dalla finestra, e per giunta accolto con la prestigiosa etichetta di attività scientifica.

E noi che ci preoccupavamo di non esagerare nel'utilizzare termini che, più propriamente, si riferiscono agli esseri umani! Quando, per semplicità, utilizzavamo il verbo pensare, nel riferirci al percorso delle istruzioni in un programma, era d'obbligo impacchettarlo tra virgolette, per intendere che il computer pensa, ma solo per modo di dire.

Anche considerando i recenti progressi tecnologici (quarzi a 20 Megahertz, processori paralleli, computer iperveloci) abbiamo sempre individuato una profonda linea di demarcazione tra l'uomo e la macchina, tra la creatività ed il semplice accostamento casuale di frasi apparentemente connesse da una struttura "intelligente".

E allora, signori scienziati (con tutto il rispetto per i veri signori) lasciate perdere le vostre sperimentazioni che sapete dove iniziano e non dove finiscono.

E queste considerazioni valgono, mi auguro, anche indipendentemente da credenze religiose o da ragioni etiche e morali; per evitare, insomma, di considerare vecchi film (e mi riferisco al Pianeta delle scimmie) non con un sorriso di sufficienza, di prammatica per un qualsiasi film di pura fantasia, ma come un manuale di comportamento utile per un Medio Evo, venturo e prossimo più che mai.

Stampante e drive

☐ I Turbo Dos rovinano i drive? Come resettare la stampante senza spegnerla e riaccenderla?
(Luigi Grasso - Somewhere)

• I vari programmi velocizzatori (cartucce, software e anche Speed Dos, Prologic e altri) costringono il drive ad effettuare letture e scritture a velocità nettamente superiori alla norma. E' quindi comprensibile il timore del nostro lettore, ma per dare una corretta risposta sarebbe necessario effettuare centinaia di prove "a fatica" su molti sistemi diversi.

Una prova del genere, in effetti, è svolta quotidianamente da migliaia di utilizzatori (tra cui il sottoscritto, che ha dotato il C/64 di Speed Dos) che, almeno finora, non hanno lamentato inconvenienti di sorta.

E' probabile un disallineamento della testina, ma tale inconveniente non sembra essere più frequente tra coloro che utilizzano sistemi velocizzatori.

Nonostante, quindi, il nostro giudizio sia ampiamente positivo sull'affidabilità e sicurezza, rivolgiamo la domanda ai nostri lettori: c'è qualcuno, tra voi, che ha notato difficoltà (di qualsiasi genere) utilizzando le utility per aumentare la velocità di operazioni I/O?

Passando alla seconda domanda, la sintassi relativa all'invio dati verso la stampante richiede tre informazioni: il numero del canale (Nc), il numero di periferica (Np, di solito 4) e l'indirizzo secondario (Is). Tipicamente l'apertura di un file è del tipo:

```
Open Nf,Np,Is
```

Non appena si accende la stampante l'indirizzo secondario vale zero ma caricando programmi commercializzati, soprattutto del tipo grafico, l'indirizzo secondario viene modificato per consentire la corretta gestione dei singoli dot (puntini da stampare), dell'interlinea, e così via.

Smettendo di utilizzare il programma in questione, molto spesso la stampante rimane settata nelle condizioni grafiche o, comunque, in modo diverso da come si trova al momento dell'accensione.

Per ripristinare le funzioni "normali" puoi tentare, in modo diretto, con...

Close 1: Open 1,4,0: Print#1: Close1

...evitando accuratamente di abbreviare il comando Print con il punto di domanda (?), pena un Syntax error.

Se, nonostante la riga suggerita (che dovrebbe ripristinare le condizioni di default), la stampante non dovesse funzionare a dovere, non rimane che spegnerla e riaccenderla.

Non temere, comunque, che accendere e spegnere apparecchi possa arrecare danno al sistema; l'importante è attendere qualche secondo prima di riaccenderlo, e di evitare di "giocare" con i vari interruttori.

Costruiamo un Modem

☐ Perchè non pubblicate lo schema di un modem di facile realizzazione?
(Mauro Nobile - Sesto S.Giovanni)

• E' vero che, con l'apporto della tecnologia moderna, i chip oggi disponibili consentono semplici realizzazioni, ma la costruzione di apparecchi affidabili dovrebbe esser riservata ai veri esperti del saldatore. La nostra non è una pubblicazione di elettronica, e se talvolta suggeriamo gli schemi di accessori, questi rimangono nell'ambito della semplicità assoluta, alla portata di (quasi) tutti i lettori.

D'altra parte ritengo che la spesa necessaria per realizzare un modem sia inferiore, al massimo, del 20% rispetto ad un analogo prodotto commercializzato, funzionante, garantito, dotato di libretto di istruzioni e, quasi sempre, di software appropriato. Risparmiare una cifra che si aggira sulle cinquantamila lire non è, quindi, consigliabile.

Rilocare

☐ Che cosa vuol dire "rilocare"?
(Antonio Pinto - Napoli)

• Prendiamo in considerazione il semplicissimo listato Basic:

```
100 input"scrivi";a
110 if a<100 then 140
120 if a>100 then 150
130 print"cento": goto100
140 print"minore": goto100
150 print"maggiore": goto100
```

Come puoi notare (a parte la sua... inutilità) vi sono, al suo interno, diversi "salti" (goto 100; then 140; then 150) ad altre parti del programma.

In Basic sono disponibili numerose utility che consentono di renumerare un programma mantenendo inalterata la logica dei "salti". In altre parole, volendo renumerare il programma di prima a partire da 235, e limitandosi a mantenere ordinati i soli numeri di linea, perverremmo al seguente listato...

```
235 input"scrivi";a
240 if a<100 then 140
245 if a>100 then 150
250 print"cento": goto100
255 print"minore": goto100
260 print"maggiore": goto100
```

...che non può funzionare a causa dell'inesistenza delle linee 100, 140 e 150. Un "vero" programma renumeratore (di serie sul C/16 e sul C/128), deve renumerare anche i cosiddetti "argomenti" delle istruzioni Goto, Gosub e Then. Una renumerazione corretta è quindi la seguente:

```
235 input"scrivi";a
240 if a<100 then 255
```

245 if a>100 then 260
250 print"cento": goto235
255 print"minore": goto235
260 print"maggiore": goto235

Anche lavorando in linguaggio macchina si ha spesso a che fare con istruzioni di salto condizionato (simili a "If...Then") o incondizionato (del tipo Goto, Gosub) e, talvolta, si presenta la necessità, per vari motivi, di riscrivere lo stesso programma in altra zona della memoria a disposizione. In casi come questo è necessario compiere un'operazione analoga alla renumerazione del Basic e che prende, più propriamente, il nome di "Rilocazione".

Purtroppo, limitandosi a lavorare con i programmi di Monitor (incorporati nel C/16 e C128), è possibile solo un'operazione simile alla renumerazione "parziale" vista prima ed è necessario, quindi, rintracciare... a mano le varie istruzioni Goto, Gosub e provvedere a modificare l'indirizzo.

Alcuni programmi specifici per lavorare in LM, fortunatamente, provvedono alla rilocazione automatica di qualsiasi programma scritto in Assembly. Come puoi intuire, però, è opportuna una profonda conoscenza del LM e della struttura del calcolatore per utilizzare con profitto i programmi-utility di cui parlo: Macro Assembler, Merlin e altri.

Da disco a nastro

□ **Posseggo solo il C/64 dotato di unità a cassetta e vorrei sapere come utilizzare programmi che richiedono espressamente l'unità a dischetti.**
(Giuseppe Valdemarin - Romans D'Isonzo)

• Chiariamo subito che i programmi commercializzati, che richiedono l'unità a dischi, non possono in alcun modo esser modificati perchè il lavoro richiesto sarebbe talmente lungo e difficile che risulterebbe più economico svolgere una qualsiasi altra attività (remunerata) e, con i soldi guadagnati, comprare il sospirato drive...

Scherzi a parte, è bene sottolineare che alcune funzioni (contabilità, archiviazione, grafica su stampante) possono esser svolte con efficienza solo se si possiede un disco, altrimenti l'operazione risulta più spedita con carta e penna.

Quando pubblichiamo programmi che richiedono il drive, ciò avviene non per motivi "promozionali" (come qualche maligno insinua) ma proprio perchè sarebbe un controsenso servirsi di un calcolatore per ottenere un'efficienza minore di quella ottenibile manualmente.

Molte delle modifiche da apportare sono state indicate (o accennate) nell'inserto del N.39 (Come gestire senza problemi un file sequenziale). Riassumerò, comunque, le informazioni principali.

L'apertura di un file sequenziale su disco è della forma...
Open Nf,Np,Is, "nome,s,r"
Open Nf,Np,Is, "nome,s,w"

...rispettivamente in lettura e in scrittura, in cui Nf, Np, Is hanno il significato illustrato in queste stesse pagine, "s" significa "sequenziale", "r" "read" (=lettura) e "w" "write" (=scrittura).

Con un registratore la sintassi è la seguente:
Open Nf,Np,Is, "nome"

in cui Np vale sempre 1, e Is vale zero (se in lettura) e 1, oppure 2, se in scrittura.

Sarà quindi sufficiente (!) rintracciare, all'interno del programma Basic che interessa, i comandi Open e apportare le dovute modifiche. I comandi Close, Print#, Get#, Input#, infatti, non richiedono modifiche (tranne che in casi particolarissimi) dal momento che la sintassi è identica sia nel caso del drive che del registratore.

Consigli

□ **Vi faccio una proposta che non dovreste rifiutare...**
(Massimiliano Vispi - Roma)

• Seguiremo, per quanto possibile, i consigli suggeriti. Altri, invece, sono già stati seguiti; in particolare:

Le routine di Toma sono state ripubblicate (e sono disponibili anche su disco nella loro versione definitiva) sul numero speciale monografico della nostra rivista "Commodore" che puoi chiedere al servizio arretrati (tel.02/84.67.348 chiedendo della signora Lucia).

La decisione di rifiutare l'invio contrassegno è stata presa sia perchè costerebbe molto di più, sia per problemi organizzativi.

Non parleremo mai di altri computer che non siano di marca Commodore: vi sono tanti lettori che si lamentano del poco spazio che dedichiamo a C/16 e C/128, figuriamoci che cosa accadrebbe se dedicassimo pagine ad altre marche!

Passando alla seconda parte della tua lettera, ti assicuro che le lettere le leggiamo tutte, e fino alla fine. Solo a questo punto, infatti, provvediamo a cestinarle...

Non funge

□ **Il programma "Alla scoperta delle tracce" pubblicato sul N.40 (di cui allego l'output della mia stampante) si blocca con la segnalazione di errore alla riga 180. Quale ne è il motivo?**
(Paolo Sabbion - Padova)

• Questa lettera mi ha fatto perdere circa due ore di tempo perchè, in effetti, il programma digitato dal lettore, a giudicare dall'ouput inviato, è identico a quello pubblicato e a quello, ovviamente, presente nel mio archivio su disco e su "Directory" (e che funzionano *perfettamente).*

Il motivo della segnalazione di errore, quindi, mi sfugge totalmente a meno che non si verifichi una delle seguenti condizioni:

a/ Il computer usato non è un C/64, come richiesto *espressamente* sulla rivista: con un C/128 (in modo 128), il programma, infatti, genera un errore del tipo lamentato; con un C/16 e un Plus/4, invece, gira normalmente, ma genera un file che non è possibile utilizzare con tali computer.

b/ Nel computer è inserita una cartuccia "strana" che altera i puntatori 63 e 64 (o i loro contenuti). Preciso che il programma pubblicato gira normalmente su un C/64 normale, su un C/64 "incorporato" in un C/128 e su un C/64 dotato di Speed Dos.

Se nessuna delle due ipotesi è verificata, ti prego di inviarmi SU DI-

SCO il listato incriminato, allo scopo di esaminarlo con maggior accuratezza: sono oltremodo curioso di scoprire il difetto!

Al contadino non far sapere...

□ Io penso che voi non pubblicate le notizie relative alle locazioni di memoria e alle routine delle Rom perchè "troppe persone saprebbero cose che solo pochi devono sapere"...
(Roberto Cariggi - Roma)

• Diamine! Mi hanno accusato di tante cose, ma addirittura di essere a capo di una setta segreta non me lo sarei mai aspettato.

Non appartengo alla P2 (al massimo alla C/64) e, lo giuro, non appena scopro qualche tecnica particolare di programmazione, o qualche utilizzo insolito di una locazione di memoria, mi affretto a divulgarla. Più di una volta, e me ne darai atto, ho spinto i lettori a comunicare le loro scoperte, e le più interessanti sono sempre state pubblicate.

Il motivo del (presunto) modesto numero di pagine dedicate all'argomento è da ricercarsi esclusivamente nella vastità dei casi possibili e nella quasi infinita applicazione delle varie routine del Kernal: pubblicarle tutte richiederebbe una rivista "infinita".

Per ciò che riguarda la taratura del drive, l'operazione è semplicissima, se svolta da un laboratorio attrezzato; vi sono in giro anche programmi (!) che provvedono a registrare correttamente la testina di lettura/scrittura. Non sempre, però, sono sufficienti e, in questi casi, è necessario consegnare il drive ad un riparatore autorizzato.

Stelle e strisce

□ Perchè, durante il caricamento (soprattutto da nastro) compaiono sul video strisce colorate?
(Maurizio Mencarini - Viareggio)

• Il sistema di registrazione originale Commodore, come è noto, risulta piuttosto lento e, di conseguenza, sono stati sviluppati molti software di supporto per sveltire le operazioni con la cassetta.

In molti casi è stato necessario modificare il sistema operativo, che gestisce, tra l'altro, il video. Si ebbero, quindi, effetti collaterali, come perdite di sincronismi, colorazione random dello schermo, alterazione del raster ed altre condizioni che, in seguito, furono sfruttate abilmente per dare un aspetto piacevole allo schermo durante il caricamento di un programma.

Come puoi intuire, però, ognuno dei numerosi Turbo in commercio ha una sua tecnica particolare di gestione dei dati.

Rimane il fatto, comunque, che le strisce di colore che noti durante un caricamento non sono altro che un "difetto" del sistema stesso!

Formattazione del 1571

□ In che modo un dischetto viene formattato con un 1571?
(Baldassarre Silvestre - Gela)
(Luigi Scarpellino - Formia)

• Un drive 1571 può essere collegato ai computer Commodore Vic/20, C/16, Plus 4, C/64, C/128; solo con quest'ultimo, però, funziona al meglio delle sue possibilità.

Il 1571, infatti, è dotato di due testine di lettura/scrittura che sono in perenne contatto con entrambe le superfici del dischetto. Se usi un qualsiasi computer (tranne che il C/128, purchè in modo 128 oppure CP/M) il drive si comporta come un normale 1541 e formatta, legge, scrive e cancella operando su una sola faccia del disco.

Con un C/128, invece, (purchè utilizzato in modo 128 oppure CP/M) impartendo il comando di formattazione si ottiene, come risultato, la formattazione di entrambe le facce del dischetto.

Se, quindi, si inserisce nel drive un dischetto già formattato in precedenza con un C/64 (o Vic/20 o C/16), il C/128 (anche se in modo 128) opera agendo su quell'unica faccia.

In conclusione è possibile ottenere un disco con 1328 blocchi solo se questo è stato formattato con un C/128 (in modo 128) collegato con un drive 1571.

In tutti gli altri casi, invece, i blocchi sono "solo" 644 e non si corre il rischio di invadere inavvertitamente la seconda faccia del disco.

Presunte inesattezze

□ In riferimento all'articolo "Relativamente utili" (CCC N.40) vi prego di puntualizzare le imprecisioni relative ai puntatori e, soprattutto, specificare che con il comando Input# non è possibile leggere più di 80 caratteri.
(Anonimo)

• Oltre a dimenticare la firma, la tua distrazione è esemplare tanto che la sottopongo all'attenzione dei lettori:
- Non precisi le... imprecisioni che dovremmo puntualizzare e di conseguenza non riesco a capire che cosa c'è da chiarire.
- Moltissime volte, in vari articoli e nelle colonne della "Posta", abbiamo avuto occasione di precisare che il comando Input# non accetta più di 80 caratteri e, ovviamente, anche il modo di porre rimedio all'inconveniente.

Ti consiglio, quindi, di leggere attentamente la nostra rivista perchè le informazioni che si cercano sono spesso "disseminate" nelle varie pagine.

Easy Script difettoso?

□ Dopo la terza o quarta copia di un file scritto con Easy Script, a volte il sistema si blocca e sono costretta a spegnere e riaccendere. Quale può esserne la causa?
(Paola Cremona - Roma)

• Uso abitualmente E/S e non mi è mai capitato un inconveniente del genere. E' probabile che la versione utilizzata sia una copia pirata... mal copiata.

Ho visto molte versioni copiate del popolare W/P; una di queste, lunga appena 51 blocchi di disco, funziona perfettamente solo con il drive mentre, servendosi del registratore, il sistema si blocca.

Siccome non precisi se il software è originale, o meno, non sono in grado di aiutarti.

Versione 2 di Ms-Dos

□ Sono soddisfatto del vostro prodotto che consente di simulare il Gw-Basic con un C/64. Quando divulgherete la seconda versione?
(Giancarlo Graziosi - Ospitaletto)

• Come promesso nella stessa confezione del software venduto in edicola, sarà disponibile una seconda versione, migliorata e più ricca di quella che possiedi.

Non penso, però, che prenderemo in considerazione la possibilità di una versione specifica per il C/128; per ciò che riguarda la possibilità di utilizzare 80 colonne con un C/64, a parte le solite considerazioni sulla nitidezza conseguibile, bisogna tener presente che il relativo software richiede aluni Kbyte per ottenere uno schermo più grande. Il numero di byte che resterebbero disponibili per l'utente rischia, quindi, di essere esiguo.

Disservizi e assistenza

□ Ho tribolato parecchio per la riparazione del mio computer... (segue accorata descrizione dei patimenti subiti).
(Attilio Peschiera - Bereguardo)

• Purtroppo, oltre che offrire la mia personale solidarietà, non mi è possibile intervenire, se non ribadendo che un centro di assistenza non può esser gestito da un qualsiasi scalzacane (con tutto il rispetto per i cani), ma richiede serietà e correttezza piuttosto particolari.

Per consolarti, però, ti posso dire che stiamo per organizzare una rubrica (con regolarità saltuaria) del tipo "La difesa del consumatore".

Per evitare inesattezze nel citare i vari casi, sarà indispensabile allegare le fotocopie della documentazione che possa testimoniare le carenze lamentate (raccomandate, bolle di consegna, garanzia, date di ingresso e di uscita, eccetera). In caso contrario non potremo pubblicare le varie lamentele onde evitare, in caso di inesattezze, denunce per diffamazione.

Elisa

□ Appassionato di programmi di simulazione, chiedo dove rintracciare il fa-

La Superstar fra le stampanti per computer é una Star!

Probabilmente, nessun'altra stampante riunisce in sè tutte le straordinarie prerogative della **NL-10**, una periferica per computer estremamente convincente nelle prestazioni e nel prezzo. **NL-10** può contare su fans in ogni settore aperto all'informatica: gestionale, organizzativo, amministrativo, sviluppo, produzione, hobbystico. Di lei gli addetti ai lavori apprezzano la semplicità d'uso e la qualità dello stampato. E' sorprendente su **NL-10** la quantità di funzioni di stampa, controllabili dall'utente tramite un pannello frontale molto sofisticato, così come la varietà dei formati di stampa e la sua enorme adattabilità a qualsiasi tipo di computer. Anche nell'affidabilità, **NL-10** darà prova di tutta la sua amicizia. Chieda al nostro rivenditore di zona una dimostrazione di Superstar **NL-10**: siamo certi che anche Lei concluderà che, **con una Star, si può andare molto lontano**!

La tua stampante

Via Gallarate, 211 20151 Milano
tel. 02/301.00.81 r.a. 301.00.91 r.a.

Per avere maggiori informazioni e l'indirizzo del rivenditore della Sua zona, ci invii il coupon allegato.

Ditta: ______ Via: ______ n° ______

Nome: ______ Cap.: ______ Città: ______

______ Tel.: ______

Il grande software made-in-Italy

LA VOCE III

Fa parlare e cantare il C64 secondo come lo programmi senza l'uso di campionatori nè sintetizzatori.
Tutte le parole o le canzoni così prodotte possono essere inserite come stringhe in altri programmi.

Lire 12.000

RAFFAELLO

Un programma per disegnare col tuo Comodore 64 col solo joystick senza Koala nè tavoletta grafica.
Tutti i disegni prodotti possono essere memorizzati ed utilizzati in altri programmi.

Lire 10.000

OROSCOPO

Fa in maniera scientifica l'oroscopo personale. Il più completo programma astrologico per Commodore 64.

Lire 12.000

COMPUTER MUSIC

Un music-editor avanzato più per un programma juke-box con 27 motivi celebri di musica classica e leggera da Arcadia a Bach, Vivaldi, Zeppelin...

Lire 12.000

GESTIONE FAMILIARE

Tre programmi su cassetta che giustificano l'aggettivo "domestico" del tuo computer:
- bilancio familiare;
- dieta equilibrata;
- scheda medica familiare.

Gira su C/64/128

Lire 12.000

BANCA DATI

Un potente data base per C/64 e Spectrum disponibile anche su disco.
L'edizione su cassetta contiene da un lato la versione C64 e dall'altro la versione Spectrum.

Lire 12.000

DICHIARAZIONE DEI REDDITI (740/S)

Programma aggiornato al 1986 per la dichiarazione dei redditi, modello semplificato. Per C64.

Disco: **Lire 24.000**
Cassetta: **Lire 16.000**

MATEMATICA FINANZIARIA

Pubblicato a puntate su Commodore (n.ri 13, 14 e 15) e su Personal Computer (n.ri 1, 2, 3 e 4) questo programma offre un vero e proprio corso completo di ragioneria su Commodore 64.
Se ne consiglia l'acquisto insieme agli arretrati delle riviste che ne illustrano l'uso ed il funzionamento.

Disco: **lire 20.000**
Cassetta: **lire 10.000**

Comodore 13, 14 e 15 e Personal Computer 1, 2, 3
Lire 21.000

ANALISI DI BILANCIO

Naturale completamento di "Matematica Finanziaria" questo programma consente di calcolare automaticamente tutti i ratio più significativi e di confrontare due bilanci dello stesso ente.
Il testo esplicativo è stato pubblicato su Personal Computer n.ri 2, 3, 4 e 5 che si consiglia di acquistare contemporaneamente.

Disco: **Lire 20.000**
Cassetta: **Lire 10.000**

Personal Computer 2, 3, 4 e 5: **Lire 12.000**

ARREDARE

Un programma professionale per ottimizzare le soluzioni d'arredamento della vostra casa.
N.B. gira solo sotto Simon's Basic.

Disco: **Lire 20.000**
Cassetta: **Lire 10.000**

GRAPHIC EXPANDER 128

Un potente programma grafico per il c 128 in modo 128.

Lire 27.000

Sì, inviatemi i seguenti software al prezzo di listino + Lire 3.000 per spese di spedizione:

- ☐ RAFFAELLO
- ☐ LA VOCE III
- ☐ OROSCOPO
- ☐ COMPUTER MUSIC
- ☐ GESTIONE FAMILIARE
- ☐ BANCA DATI
- ☐ MATEMATICA FINANZIARIA/DISCO
- ☐ MATEMATICA FINANZIARIA/CASS.
- ☐ MATEMATICA FINANZIARIA/RIVISTE
- ☐ ANALISI DI BILANCIO/DISCO
- ☐ ANALISI DI BILANCIO/CASS.
- ☐ ANALISI DI BILANCIO/RIVISTE
- ☐ DICHIARAZIONE DEI REDDITI/DISCO
- ☐ DICHIARAZIONE DEI REDDITI/CASS.
- ☐ ARREDARE/DISCO
- ☐ ARREDARE/CASSETTA
- ☐ GRAPHIC EXPANDER/DISCO

Valore complessivo: Lire..

Su tale importo mi praticherete lo sconto del 10% in quanto abbonato a ☐ *Commodore Computer Club* ☐ *Personal Computer* ☐ *Computer* ☐ *VR Videoregistrare*. Pertanto vi invio la somma soltanto di lire..

☐ Desiderando ricevere le copie ordinate con la massima urgenza, accludo assegno bancario n.ro..
Banca.. per lire.. voi intestato.

☐ Contentandomi dei normali tempi postali ho inviato oggi stesso l'importo di lire.......................... a mezzo C/C postale N. 37952207 intestato a Systems Editoriale - V.le Famagosta, 75 - 20142 Milano.

Ritagliare e spedire in busta chiusa regolarmente affrancata a Systems Editoriale - V.le Famagosta, 75 - 20142 Milano.

Nome ....
via .... **N.ro.** .... **telefono** ....
CAP .... **Città** ....

Entra nel grande Club

Fin dallo sbarco in Italia della Commodore ***Commodore Computer Club*** è il punto di riferimento di tutti gli utenti di C/64, Vic 20, C/16, Plus 4 ed ora di PC 10/20 ed Amiga.

Articoli didattici, recensioni e programmi istruttivi ed a basso costo hanno fatto di *Commodore Computer Club* la prima rivista italiana d'informatica.

Ma, per i lettori, Commodore Computer Club non è solo rivista: è consulenza telefonica gratuita, software originale pubblicato a latere dalla stessa casa editrice, un ponte verso l'informatica "maggiore" anche attraverso la collaborazione con le riviste sorelle *"Personal Computer"* e *"Computer"*.

E' per questa ragione che, anno dopo anno, aumenta il numero dei lettori che preferiscono ricevere la rivista in abbonamento invece di acquistarla in edicola. Ad essi l'editore riserva una serie di vantaggi esclusivi come:

- **un libro in omaggio** da scegliere tra i titoli disponibili della collana I libri di Systems*;
- **l'uso di una linea telefonica speciale** per richieste di consigli, e consulenza, il cui numero e le modalità d'uso verranno comunicate in forma riservata alla ricezione dell'abbonamento;
- **un canone annuo particolarmente interessante** di lire 40.000 per 11 fascicoli di Commodore Computer Club e di lire 35.000 per 11 fascicoli di Personal Computer;
- **l'esclusivo canone cumulativo** di lire 65.000 per 11 fascicoli di Commodore Computer Club ed 11 di Personal Computer;
- **uno sconto del 10%** su tutti gli acquisti per corrispondenza dei prodotti software su disco o cassetta, fascicoli arretrati o libri della Systems senza limiti di quantità.

** I titoli disponibili sono quelli reclamizzati sull'apposita pagina pubblicitaria "La libreria di Systems".*

Inviatemi in omaggio il volume della collana i libri di Systems...

Registrate oggi stesso il mio abbonamento a: ☐ Commodore Computer Club (Lire 40.000)
☐ Commodore Computer Club+Personal Computer (Lire 65.000)

☐ Desiderando ricevere le copie ordinate con la massima urgenza, accludo assegno bancario n.ro..................
Banca.................................... per lire.................................... voi intestato.

☐ Contentandomi dei normali tempi postali ho inviato oggi stesso l'importo di lire .... a mezzo C/C postale N. 37952207 intestato a Systems Editoriale - V.le Famagosta, 75 - 20142 Milano.

Nome ....
via .... **N.ro.** .... **telefono** ....
CAP .... **Città** ....

Ritagliare e spedire in busta chiusa regolarmente affrancata a Systems Editoriale - V.le Famagosta, 75 - 20142 Milano.

moso "Elisa" in cui il computer impersona lo psicologo che visita il paziente-utente.
(Affezionato lettore di Caivano)

• Vi sono numerosissime versioni del programma che tu menzioni ed una di queste, addirittura, è inserita in uno dei dischetti di "Directory" che vendiamo per corrispondenza. In questo programma, addirittura, il C/64 ti PARLA ricorrendo ad un sintetizzatore vocale.

Riproduzioni sonore

☐ **Come si può registrare, e riprodurre, un segnale di bassa frequenza con un C/64?**
(Stefano Buonomo - Caiazzo)

• Un segnale di bassa frequenza, del tipo che normalmente si può osservare con un oscilloscopio, può appartenere ad una infinità di "forme" che dipendono dalla frequenza, dall'ampiezza, dalla presenza di eventuali armoniche e da tantissimi altri parametri.

Per evidenziare una vasta gamma di frequenze è indispensabile, appunto, un oscilloscopio, vale a dire uno strumento che "tratta" la frequenza stessa in tempo reale. Mi spiego meglio: in un oscilloscopio, di solito, un segnale in bassa frequenza viene amplificato e inviato immediatamente ad un circuito elettronico in grado di modificare il percorso di un pennello di elettroni. Il risultato consiste, appunto, nella traccia visibile sullo schermo fluorescente dello strumento.

Utilizzando un computer la faccenda si complica notevolmente: poichè un calcolatore può trattare solo "zeri" e "uni" è indispensabile che il segnale B.F. venga dapprima esaminato e "tradotto" in un segnale il cui livello picco-picco sia, appunto, 0-1 (cioè 0 oppure 5 volt). In seguito è necessario suddividere il segnale in esame in modo da codificarlo per consentire di passare da una grandezza analogica alla corrispondente digitale. Ma non è finita: l'informazione, a questo punto, deve essere elaborata e inserita nella memoria del computer in byte successivi. Un'operazione inversa deve esser compiuta per riprodure la stessa frequenza.

Come puoi intuire, quindi, il trattamento di un qualsiasi segnale B.F. è piuttosto difficile e diventa addirittura impossibile se si pretende di utilizzare un home computer come affidabile strumento di misura: un frequenzimetro e un oscilloscopio sono sicuramente più attendibili di un C/64 dotato di software idoneo.

Nonostante le difficoltà accennate, tuttavia, il chip audio del C/64 è abilmente sfruttato per campionare suoni che possono esser raccolti da un comune microfono. Ma di questo, come sai, abbiamo già parlato sul N. 40 della nostra rivista.

Differenze

☐ **Che differenza c'è tra Load e Verify? E tra Tab(X) e Spc(X)?**
(Claudio Daffra - Pavia)

• Il comando Load è indispensabile per caricare un programma da supporto magnetico e la sintassi cambia a seconda della periferica usata: è sufficiente un semplice "Load" per caricare il primo programma presente su nastro, oppure Load "Pippo" per caricare, sempre da nastro, il programma dotato di nome "Pippo". Con il disco, invece, è indispensabile indicare sia il nome che la periferica: Load"Pippo",8

Con Verify, invece, è possibile verificare che il programma presente nella memoria Ram del computer sia uguale a quello presente su supporto magnetico. Ad esempio, il comando...

Verify "pippo",8

...permette di verificare eventuali differenze tra il programma presente in memoria e quello, di nome "pippo", memorizzato sul dischetto. In caso di differenze, anche di un solo byte, compare il messaggio "Verify error", altrimenti un semplice "OK".

Per megio comprendere la differenza tra Tab e Spc, ti consiglio di digitare (e studiare) il seguente programma:

```
100 print tab(10)"primo";
110 print tab(10)"secondo"
120 print
130 print spc(10)"primo";
140 print spc(10)"secondo"
```

Come puoi notare (attento alla presenza dei caratteri di punto e virgola) si tratta di un doppione di istruzioni Print riferite a "Tab" e "Spc".

Facendo girare il programma, la prima volta le due parole "Primo" e "Secondo" appaiono attaccate tra loro; con le righe 130 e 140, invece, appaiono separate tra loro di dieci spazi.

Il comando Tab (utilizzabile, come Spc, sempre e solo se preceduto da Print) funziona nel modo seguente:

Se il messaggio successivo a "Primo" (cioè: "secondo") si trova ad una distanza dal bordo sinistro superiore a quella indicata tra parentesi, allora il messaggio "Secondo" viene, appunto, visualizzato a tale distanza. Se, invece, il cursore, al momento di eseguire il comando, si trova già ad una distanza superiore a quella indicata tra parentesi, il Tab viene ignorato ed il messaggio successivo a Tab viene stampato subito dopo quello precedente.

Con Spc, invece, il messaggio presente dopo viene sempre visualizzato alla distanza indicata tra parentesi.

Per comprendere meglio il funzionamento del programma proposto, prova a sostituire, nelle parentesi di riga 100, il valore 10 con 1, 4 e altri valori.

Mal di fegato

☐ **Sono il direttore di una rivista di informatica. Alcuni lettori che possiedono il C/128 o il C/16 si lamentano perchè giudicano eccessivo lo spazio dedicato al C/64; i possessori di C/64, viceversa, considerano uno "spreco" lo spazio dedicato al C/128. Mi sta venendo il mal di fegato...**
(Alessandro de Simone - Milano)

• In casi come questo può essere consigliato l'amaro medicinale Giuliani, ma attenzione: è un medicinale, usare con cautela.

Risposte rapide

Speed Dos 128

Vi sono in giro molte versioni di Speed Dos, diverse l'una dall'altra anche in modo sostanziale: ti consiglio di contattare le varie Ditte che le commercializzano per sapere se è possibile applicarle al tuo calcolatore.

(Marco Crisafulli - Somewhere)

Centronics

Una qualsiasi stampante che disponga dell'interfaccia Rs-232, IEE-488 oppure Centronics, può IN TEORIA, funzionare correttamente con il tuo C/64. Tieni presente, però, che devi comprare anche l'interfaccia per il calcolatore e il cavo idoneo. Nella quasi totalità dei casi, purtroppo, una stampante non esplicitamente definita "Mps-803 compatibile", può manifestare incompatibilità, soprattutto se adoperata con programmi grafici.
(Pasquale Virgilio - Andria)
(Pasquale Nobile - Taviano)

Linguaggio Macchina

Il numero monografico di Commodore dedicato al linguaggio macchina e alla grafica si riferisce al LM del microprocessore 6502. Gli altri argomenti (interrupt, raster, eccetera) sono invece specifici dei computer su cui il micro è montato. Stiamo preparando, comunque, un secondo numero monografico su tali argomenti e ti promettiamo che i vari inconvenienti lamentati (tra cui la mancanza della tabella, pubblicata, poi, su CCC N.40) non si ripeteranno.
(Marco Neri - S.Giorgio a Cremano)

Wordstar 128

Purtroppo non sono in grado di stabilire le cause del malfunzionamento del tuo C/128 con il word processor Wordstar; penso, tuttavia, che il motivo sia da addebitare alla stampante che hai collegato, una Panasonic Kx-P1092, che, come quasi tutte le stampanti non Commodore compatibili, crea questi (e altri) problemi.
(Mario Leoncini - Siena)

Graphic Expander 128

Il nostro prodotto, Graphic Expander C128, idoneo per gestire la grafica in alta risoluzione con il C/128 in modo 80 colonne, è venduto corredato da un demo che, studiato opportunamente, consente di apprezzare le potenzialità dei nuovi comandi. Oltre alla sintassi delle istruzioni aggiuntive (pubblicate su CCC N.35), infatti, solo un programma dimostrativo è in grado di illustrarne le molteplici applicazioni. Per ciò che riguarda la cassetta non funzionante, rimandacela per la sostituzione.
(Carlo Rando - Cassano Jonio)

Directory

Per problemi organizzativi non ci è possibile, almeno per ora, distribuire il dischetto "Directory" nelle edicole. L'idea, alla quale stiamo pensando da tempo, non è comunque del tutto accantonata.
(Maurizio Briganti - Roma)

Ci dispiace, ma...

Il programma che ci hai inviato non risponde alle nostre esigenze. Ti consiglio di telefonare prima di inviare un lavoro in modo da verificare se vale la pena spedirlo: una telefonata in teleselezione (effettuabile, magari, di pomeriggio) costa molto meno del dischetto e dell'affrancatura di un pacchetto raccomandato.
(Ferdinando Andretta - Caivano)

Turbotape e videogame

Non sempre un'utility in linguaggio macchina (come, appunto, il turbo tape) può "convivere" con altri programmi, specie se sono scritti anch'essi in LM; forse è questo il motivo per cui alcuni programmi, caricati con il T/t in tuo possesso, mandano in Crash il sistema.
(Antonio Miccoli - Lecce)

I tasti del C/128

La tastiera del C/128 in tuo possesso consente, a seconda se premi, o meno, il tasto Ascii/CC, di utilizzare la tastiera stessa in modo Qwerty oppure Qzerty (i primi sei tasti della seconda fila) che sono i due principali tipi di tastiere esistenti. Puoi scegliere l'una o l'altra indifferentemente, ma solo se sei in modo 128. In modo 64 la tastiera funziona sempre come se fosse Qwerty. Vi sono molte stampanti 803-compatibili ed un loro elenco completo sarebbe impossibile, specie se "slegato" da considerazioni sull'affidabilità degli importatori.
(Samuele Faraon - Somewhere)

Mouse 1

Il mouse da noi proposto sul N.34 è costituito da due potenziometri a slitta; quello della Commodore è invece di tipo elettromeccanico e ciò spiega l'incompatibilità del nostro mouse con il programma Geos.
(Alessandro Villamaina - Genova)

Mouse 2

Se sei esperto nei montaggi elettronici, la costruzione del simulatore di mouse non dovrebbe rappresentare un pericolo. In caso contrario ti sconsiglio di metterti all'opera.
(Franco Berta - Montalenghe)

Agenda

Non abbiamo pubblicato un programma per la gestione di un'agenda. In commercio ve ne sono molti e, al limite, anche un Data Base, se ben programmato, può simularne le funzioni.
(Stefano Giannini - Firenze)

Syntax Error

Il tuo computer segnala errore perchè il listato che hai digitato presuppone, come chiaramente spiegato nell'articolo che citi, il preventivo caricamento delle routine grafiche di Toma.
(Maurizio Guidi - Calcinaia)

Compatibilità

Solo la Ditta che tu citi può comunicarti se l'apparecchio posto in vendita è compatibile, o meno, con il tuo C/16. Scrivi (o telefona) alla Ditta in questione: noi non possiamo sapere tutte le caratteristiche dei numerosissimi accessori commercializzati.
(Giovanni Turturro - Giovinazzo)

Come procurarsi

I nostri prodotti (riviste, cassette e, tra breve, anche dischi) sono normalmente distribuiti in edicola. Per procurarsi gli arretrati è sufficiente leggere le nostre pagine appositamente dedicate al servizio offerto. Se, tra i nostri annunci, manca il prodotto che ti interessa, telefona al servizio arretrati per sapere se è disponibile (Tel.02/84.67.348, chiedendo della signora Lucia).

Cavetto difettoso

Quasi certamente il cavetto che collega il drive al computer è da cambiare: forse è presente qualche corto circuito interno oppure non è ben inserito nella sua sede.
(Enrico Zogno - Alpignano)

Amiga e Ms-Dos

I nuovi modelli di Amiga sembrerebbero (il condizionale è d'obbligo) compatibili con lo standard Ms-Dos e non richiedono il (costoso) accessorio Sidecar, "accontentandosi" di una scheda aggiuntiva. Non appena ne sapremo di più, ne parleremo.
(Alessandro Rubino - Napoli)

Purchè funzioni!

Ovvero: come attivare, da LM, le funzioni matematiche del C/64

di Giancarlo Mariani

Quanti di voi, programmando in LM, si sono trovati di fronte al problema di far calcolare al C/64 il valore di una funzione matematica o trigonometrica?

In Basic è molto facile: basta impartire un banale Print Funz(arg) ed il computer provvede.

In LM, invece, il problema non è semplice anche possedendo una certa conoscenza del 6502 e avendo a disposizione il disassemblato commentato del S.O. del computer.

Le routine che calcolano i valori delle funzioni sono già allocate, come è intuitivo, nelle Rom del S.O. e non sarà quindi necessario riscriverle; basterà trovare un modo per "passare" loro l'argomento e per leggere il risultato. Quest'ultimo risulta essere l'unico problema da risolvere, perchè per il resto basta un salto alla routine appropriata.... ed il gioco è fatto.

E' bene sottolineare che le funzioni trattate in questo articolo sono tutte ad un solo argomento (del tipo Sin, Log, Tan, eccetera) e non del tipo And, Or e simili.

Tali funzioni accettano, come ingresso, un numero reale in virgola mobile (detta Floating Point o, più semplicemente, FLP) e forniscono un risultato anch'esso di tipo FLP. Poichè i numeri FLP sono piuttosto "duri" da trattare in LM (essendo formati da ben 5 byte), ho trovato il modo di semplificarne la gestione, trasformandoli in numeri interi a due byte, nella consueta forma byte basso-byte alto, notazione ben più comoda da trattare in LM.

Si intuisce che, pur perdendo i decimali e limitando il "range" dei risultati, si ottiene il vantaggio di una notevole semplificazione delle operazioni.

Ma andiamo per gradi, cercando di capire prima come si fa a costruire una breve routine in LM che permetta di determinare il valore di una funzione. Ci riferiremo, nel prossimo esempio, ad una funzione scelta a caso tra quelle presenti: la radice quadrata (SQR).

Un primo esempio

Il segmento di S.O. che implementa tale funzione è posto a partire dall'indirizzo esadecimale $BF71. Per utilizzarlo basterà trascrivere, nell'accumulatore FLP n.1 (che è una particolare zona del C/64, usata come memoria di transito per i numeri in virgola mobile) l'argomento della stessa radice, richiamare, tramite JSR, la routine di SQR, ed infine leggere il risultato, sempre dal FLP ACC#1.

Provate, servendovi di un monitor di LM (o del caricatore Basic qui riportato), a digitare il brevissimo programmino, partendo da un indirizzo qualsiasi (ad esempio 8192, $2000):

```
$2000:
JSR $AEFD;  controlla virgola
JSR $AD8A;  n. reale dopo la virg.
JSR $BF71;  Esegue SQR
JMP $AABC;  Stampa n. FLP
```

Oppure, in Basic, da 49152:

```
100 data 32,253,174,32,138,173,32,
    113,191,76,188,170
110 for i=49152 to 49152+11: read
    x:'pokei,x: next
120 input"numero";n
130 sys 49152,n: goto 120
```

Dopo aver impartito il comando Sys, insomma, sullo schermo verrà stampato il numero reale che rappresenta il risultato della radice eseguita (verificate, magari, tramite un banale: Print Sqr (arg)).

Tentando di attribuire alla variabile "N" un valore negativo, compare il familiare "Illegal quantity".

Vediamo di capire come funziona la routine: Il primo JSR (JSR $AEFD) serve solo per controllare la presenza della virgola dopo il comando SYS. La routine S.O. posta in $AD8A, che richiamiamo con il secondo JSR, serve a "catturare" il numero reale scritto dopo la virgola, a trasformarlo in FLP, ed a trasferirlo nel FLP ACC#1.

Il calcolo della radice viene eseguito, come spiegato prima, dal JSR $BF71 che provvede anche a trasferire il risultato (sempre in FLP) nell'ACC#1.

L'ultimo JMP (JMP $AABC) richiama una parte del comando Basic "Print", che stampa il risultato in forma di numero reale.

La spiegazione dell'uso delle funzioni in LM potrebbe esaurirsi qua; ma dal momento che è impensabile trattare numeri in FLP direttamente in LM (senza cioè l'ausilio del Basic) bisognerà convertire questi numeri in una forma più direttamente trattabile dal LM.

Direttamente in LM

La forma più semplice per trattare i numeri in LM è la solita "byte basso-byte alto", anche se, purtroppo, limita il "range" degli argomenti e dei risultati dell'intervallo 0-65535 ed elimina i decimali, a patto che vi accontentiate di ritenerli poco frequenti lavorando in LM.

Le routine per il calcolo delle funzioni accettano, in ingresso, un numero FLP; bisognerà quindi convertire il numero intero in numero FLP. Esaminiamo una possibile modifica da apportare al programmino visto prima:

```
$2000:
JSR $AEFD;    Virgola
JSR $AD8A;    Prende il n.
              compreso tra
JSR $B7F7;    0 e 65535 e lo
              mette in $14/$15
LDA $14  ;    Trasferisce num.
              in A/Y
LDY $15
JSR $B391;    Trasforma num da
              int (A=lo/Y=hi) a FLP
JSR $BF71;    Calcolo SQR
JMP $AABC;    Stampa FLP
```

Tralasciando la spiegazione del JSR $AEFD, si notano due JSR posti subito dopo. Questi salti al S.O. servono per leggere il numero intero compreso tra 0 e 65535 posto dopo la virgola ed a metterlo nella forma byte basso byte alto nelle locazioni $14/$15.

La routine posta a $B391 trasforma il numero intero contenuto in A (lo) e Y (hi) in numero FLP e deposita questo risultato nel FLP ACC#1.

A questo punto il programma può continuare come il precedente, ossia con il calcolo della radice quadrata (JSR $B7F1) e con la stampa del risultato ($AABC).

Naturalmente, oltre ad avere un numero intero come ingresso, per alcuni programmi può necessitare lo stesso tipo di risultato. Per fare una cosa del genere, potremo ovviamente tener valida tutta la prima parte del programma visto prima, limitandoci ad aggiungere la parte che occorre per trasformare il risultato della radice da FLP a intero.

Vediamo che forma può avere una routinetta adatta allo scopo:

```
$2000:
JSR $AEFD
JSR $AD8A
JSR $B7F7
LDA $14
LDY $15
JSR $B391
JSR $BF71
JSR $B7F7;    Converte n. in
              FLP ACC#1 a n.
              intero in $14/$15
LDX $14;      Trasferisce
              num. in X/A
LDA $15
JMP $BDCD;    Stampa intero
              cont. in X (lo)
              e A (hi)
```

La prima parte (fino al JSR $BF71) non è commentata (è uguale a quella di prima). Poichè il risultato è un numero in virgola mobile, bisognerà richiamare la routine del S.O. (posta a $B7F7) che provvede a trasformare il numero FLP, contenuto nel FLP ACC#1, in numero intero a 16 bit, contenuto nelle locazioni $14 (lo) e $15 (hi).

Il JMP $BDCD provvede a stampare su video il numero intero contenuto in X (lo) e A (hi).

Tabella degli indirizzi di funzioni disponibili nel C/64

```
SQR  = $BF71
LOG  = $B9EA
EXP  = $BFED
COS  = $E264
SIN  = $E26B
TAN  = $E2B4
ATN  = $E30E
RND  = $E097
```

Nella tabella riportata sono indicati gli indirizzi di partenza delle routine matematiche e trigonometriche del C/64. Per utilizzarle sarà sufficiente sostituire l'indirizzo $BF71, relativo alla radice quadrata, con quello desiserato.

SCHEDA TECNICA

Osservazioni ed esempi sull'utilizzo di alcune routine del Sistema Operativo.

Note valide per computer C/64 e non adattabili ad altri computer Commodore

Consigliato a chi già conosce gli argomenti base del Linguaggio Macchina.

Cara Amiga, ti scrivo...

Come scrivere file di testo con l'Amiga, limitandosi ad usare CLI; ed una chiacchierata sull'Amiga Basic

di Luigi Callegari

Un file si dice di tipo "BATCH" quando contiene un insieme di comandi di AmigaDOS scritti esattamente come se fossero digitati da tastiera in modo CLI.

Non tutti sanno che quando si inserisce il disco Workbench, dopo la richiesta del computer, viene automaticamente ricercata, nella directory denominata "s", un file "batch" in grado di inizializzare il sistema.

Sul dischetto originale esiste (a meno che non abbiate combinato pasticci...) la directory "s" in cui è presente un file chiamato "startup-sequence": questo contiene, appunto, una lista di comandi che "configurano" il sistema. Vedremo in futuro come sia opportuno manipolare i contenuti di tale file per ottenere la autoconfigurazione del vostro sistema quando lo doterete di nuove periferiche, come ad esempio un secondo disk drive, un hard-disk, o una interfaccia video.

Esiste, in AmigaDOS, un comando che visualizza un file così come è presente sul disco, a patto che i byte che lo compongono siano codici Ascii che significhino qualcosa; si tratta di "Type", la cui forma sintattica completa è la seguente:

TYPE (FROM) nomeorig ((TO) nomedest) (OPT N o OPT H)

La parola From, come indicano le parentesi, è opzionale se il nome che

segue Type è quello del file sorgente; se specificata deve essere seguita dal nome del file interessato.

La parola TO segue la stessa regola, per il nome del file o della periferica (stampante, interfaccia seriale, eccetera) al quale deve essere inviata l'uscita di Type; ovvero è opzionale se il secondo nome specificato è quello della destinazione; se specificato deve essere seguito dal file/periferica di destinazione. Per default, se non si specifica nulla, l'uscita è inviata alla finestra video corrente.

Le due opzioni possibili sono OPT N e OPT H.

Se si accoda al comando la prima di esse, ogni linea, in Output, viene preceduta da un numero progressivo, come se fosse un listato Basic del C/64. Si noti che Type considera terminata una linea se è presente un codice Ascii di "line-feed" (dec.10 Hex.0A).

Specificando l'opzione OPT H, invece, si ottiene la visualizzazione del file in codice esadecimale, seguita dai corrispondenti caratteri Ascii. Quest'ultima forma sintattica è indispensabile per esaminare file non di testo che creerebbero confusione sul video se non, addirittura, l'inchiodamento della macchina.

Le due opzioni, come è intuitivo, si escludono vicendevolmente.

Per esaminare il file batch di inizializzazione di Workbench, presente sul dischetto in vostro possesso, provate con:

Type workbench:s/startup-sequence
Type workbench:s/startup-sequence OPT H
Type workbench:s/startup-sequence OPT N

L'Editor

Come in tutti i sistemi operativi che si rispettino, AmigaDOS prevede un editor di file, ovvero un programma che permette di creare un file da tastiera.

Per la precisione esistono due editor: uno chiamato "ED", di tipo "a tutto schermo", ed uno chiamato EDIT "orientato alla singola linea", meno immediato da usare, ma per certi versi più comodo.

Ciascuno dei due possiede numerosissime opzioni, tali da renderli molto simili ad un vero programma di videoscrittura. Ad esempio è possibile fissare tabulatori, spostare, cancellare, copiare parti di testo, convertire minuscole in maiuscole, ricercare e/o sostituire parole, eccetera.

Tali editor risultano indispensabili per creare file "batch", oppure per modificare quelli presenti su dischi commerciali o, addirittura, sul nostro Workbench.

Parliamo prima dell'editor a tutto schermo. Per attivarlo da CLI si usa un semplice:

ED (FROM) nomefile ((SIZE) numero)

Le parole FROM e SIZE sono opzionali, ma da specificarsi se si vuole indicare prima il parametro "numero" e poi "nomefile". Il nome del file può riferirsi ad un file esistente sul disco, oppure no.

Nel primo caso il file viene caricato pronto per essere modificato, nel secondo caso verrà creato "ex novo", nè più nè meno, insomma, di ciò che accade con il CP/M o con l'Ms/Dos.

Il "numero" è opzionale, come indicano le parentesi, e si riferisce al numero di byte da usarsi come "buffer" per l'edizione. Nel caso si stia modificando un file, il sistema provvede a calcolare la quantità necessaria; ma nel caso si vogliano aggiungere molti caratteri ad un file da modificare, oppure si voglia creare un file piuttosto voluminoso, è bene indicare un numero di byte abbondante, per sicurezza.

Ad esempio, per creare un file chiamato "mary" sul dischetto inserito nel drive interno, nella directory "s", con un massimo di circa 1000 caratteri, si potrebbe usare:

ED s/mary 1000

Una volta entrati nell'editore, si può creare il testo Ascii desiderato. Si ricordi che il tasto di Return inserisce un "line-feed", che i tasti di cursore agiscono sull'intero schermo e che esistono due modi per impartire comandi: un modo DIRETTO ed un modo ESTESO.

Nel primo si agisce semplicemente premendo una combinazione di due tasti oppure un tasto dedicato (BACKARROW per cancellare il carattere precedente, DEL per il carattere presente sotto il cursore).

Il secondo modo si ottiene pigiando prima il tasto ESC; in questo caso appare un asterisco sull'ultima linea dello schermo ed è possibile digitare da tastiera il comando voluto.

Per solo dovere di cronaca, e di completezza, elenchiamo qui di seguito, pur se in modo succinto, i comandi disponibili.

Questi, come per qualsiasi Editor presente sui numerosi Sistemi Operativi, risentono dell'arcaicità di un simile modo di operare. I comandi, infatti, sono talmente numerosi (e soprattutto difficili da ricordare) che è piuttosto raro, al giorno d'oggi, trovare chi, per scrivere un file, ricorra all'Edit di sistema piuttosto che a programmi specifici, molto vicini ai comodi Word processor, che consentano una più agevole stesura di un qualsiasi testo.

Ecco ora i numerosi comandi, disponibili con ED, che si possono ottenere mediante la pressione successiva di alcuni tasti:

Modo diretto:

- Tasti cursore: spostano sul video il cursore
- Tab, oppure Ctrl-I : spostano in avanti di una posizione il cursore.
- Ctrl U: sposta il testo verso l'alto
- Ctrl D: sposta il testo verso il basso
- Ctrl E: sposta il cursore verso la prima/ultima linea
- Ctrl T: sposta il cursore all'inizio della parola successiva
- Ctrl R: sposta il cursore indietro di una parola.
- Ctrl: sposta il cursore all'inizio/fine della linea attuale
- Backspace (oppure Ctrl H): cancella il carattere presente prima del cursore
- DEL: cancella il carattere posto sotto il cursore
- Ctrl A: inserisce una linea nuova, vuota, sotto quella attuale
- Ctrl B: sopprime la linea corrente, ove si trova il cursore
- Ctrl O: cancella la parola o lo spazio dopo il cursore
- Ctrl Y: cancella una linea dal cursore sino alla fine
- Ctrl F: commuta in maiuscolo (minuscolo) il carattere che si trova sotto il cursore
- Ctrl V: verifica lo schermo
- Ctrl G: ripete l'ultimo comando del modo dei comandi estesi
- Ctrl V: verifica la correttezza dello schermo
- ESC o Ctrl: attiva il modo dei comandi estesi

Questi sono i comandi disponibili nel modo "extended":

- Cs: sposta il cursore all'inizio della riga
- Ce: sposta il cursore alla fine della riga
- Cl: sposta il cursore di una posizione verso sinistra
- Cs: sposta il cursore di una posizione verso destra
- N: pone il cursore all'inizio della linea successiva
- P: pone il cursore all'inizio della linea precedente
- T: pone il cursore all'inizio (Top) del file
- B: pone il cursore alla fine (Bottom) del file
- D: sopprime una riga intera
- Dc: elimina il carattere sotto il cursore
- I/k: inserisce la linea "k" sopra la riga attuale
- A/k: inserisce la linea "k" sotto la riga attuale
- S: spezza in due la riga alla posizione del cursore
- J: unisce la riga attuale e la susseguente
- Slc: fissa il margine sinistro in posizione "c"
- Src: fissa il margine destro in posizione "c"
- Ex: aumenta il margine sinistro comunque sia
- Stn: fissa ad "n" la distanza dei tabulatori
- Sh: visualizza lo stato (margini, tabulatori...)
- U: cancella l'ultima variazione fatta
- Sa/n/: salva il file con nome "n" sul disco
- If/n/: inserisce il file "n" a partire dal punto in cui si trova il cursore
- ; (punto e virgola): permette un altro comando sulla stessa linea
- () (parentesi): ingloba i comandi per ripeterli
- Rp: ripete i comandi sinchè non incorre in errore
- Q: esce dall'editor senza salvare il file
- X: esce dall'editor salvando il file con le variazioni eventualmente apportate
- #: ripete il comando per un numero "#" di volte

Esistono anche alcuni comandi del modo "extended" che operano su di un blocco di linee per volta.

- Bs: fissa l'inizio di un blocco
- Be: fissa il termine di un blocco
- Db: cancella il blocco prima delimitato
- Ib: inserisce il blocco prima delimitato
- Sb: mostra sullo schermo il blocco
- Wb/n/: inserisce il blocco nel file di nome "n"

L'Editor di linea

Viene attivato da CLI con un comando del tipo:

EDIT FROM orig TO dest WITH coma VER dire OPT opz

Le parole sono opzionali solo quando i parametri vengono specificati ordinatamente come indicati.

"Orig" rappresenta il nome del file da elaborare ed è, ovviamente, tassativo.

"Dest" rappresenta il nome del file dove memorizzare il frutto del nostro lavoro.

"Coma" indica il nome di un file che viene usato come ingresso alla linea comandi dell'editor e che deve contenere una serie di parole accettabili come tali.

"Dire" permette di stabilire il device logico o il file ove inviare i messagi di errore eventualmente prodotti dall'editor.

La parola "OPT" può essere seguita da "W" o "P", a loro volta seguite da un numero: dopo "W" si deve specificare la lunghezza massima della linea, dopo la "P" il numero massimo di linee.

Quando viene attivato per la prima volta, l'editor comincia ad elaborare la prima linea, memorizzandola nel cosiddetto "buffer" di edizione e ne permette l'elaborazione tenendo conto della serie di comandi elencati di seguito.

Le varie linee, a mano a mano che procede l'elaborazione, vengono puntate dal cosiddetto "current line marker", che può essere controllato da alcuni dei comandi dell'editor: spostando il puntatore dalla linea la si memorizza nel buffer di uscita, pronte per essere effettivamente memorizzate quando si esce dall'editor.

Ecco, di seguito i comandi disponibili:

• (minore) <: sposta a sinistra il puntatore di carattere
• (maggiore) >: sposta a destra il puntatore di carattere
• PR: resetta il puntatore di carattere
• (percento) %: trasforma il carattere puntato in maiuscolo
• (dollaro) $: trasforma il carattere puntato in maiuscolo
• (meno)-: trasforma il carattere puntato in uno spazio
• (cancelletto) #: elimina il carattere puntato
• SHG: visualizza tutte le informazioni disponibili
• PA/s/: sposta il puntatore dopo la stringa "s"
• PB/s/: sposta il puntatore prima della stringa "s"
• N: avanza di una riga
• P: retrocede di una riga
• Ml: muove alla linea "l"
• Tn: emette "n" linee
• TP: emette le linee precedenti
• T: emette le linee successive
• TLn: emette "n" linee precedute dal numero di linea
• (punto di domanda) ?: verifica la linea corrente
• (punto esclamativo) !: verifica la linea corrente ed i codici
• V+/V-: attiva/disattiva la verifica di linea
• TR-/TR+: trascura/visualizza la linea tra le righe
• I/l: inserisce il testo prima della linea corrente
• I• o R•: inserisce il testo alla fine del file
• Ir: inserisce il testo prima della riga "r"
• Z/c/: cambia al carattere "c" il terminatore di input
• I/nf/: inserisce il file "nf" prima della corrente riga
• I•/nf/: inserisce il file "nf" dopo la fine del file corrente
• Ir/nf/: inserice il file /nf/ prima della riga numero "r"
• R/R: rimpiazza la riga corrente con il testo inserito
• Rrl r2: rimpiazza la riga numero "rl" con la "r2"
• (eguale) =l: rinumera tutte le linee assegnando "l" alla corrente
• Drl r2: elimina le linee tra la numero "rl" e "r2"
• DTB/st: elimina il testo prima della stringa "st"
• DTA/st: elimina il testo dopo la stringa "st"
• DF/st: elimina fino a "st"
• SB/st/: spezza la linea prima della stringa "st"
• SA/st/: spezza la linea dopo la stringa "st"
• CL: ricongiunge una linea spezzata
• CL/st: unisce la riga corrente con la stringa "st"
• GA/s/s2/: inserisce completamente la stringa "s2" dopo "s"
• GB/s/s2: inserisce completamente la stringa "s2" prima di "s"
• CGn: cancella completamente l'operazione "n"
• CF: cancella tutte le operazioni
• F/str/: ricerca nel file la stringa "str"
• BF/str/: ricerca una stringa "str" all'indietro
• E/sl/s2/: sostituisce la stringa "s2" con la "sl"
• B/sl/s2/: inserisce la stringa "s2" prima di "sl"
• A/sl/s2: inserisce la stringa "s2" dopo la stringa "sl"
• W: continua l'elaborazione attraverso il file sorgente
• Q: esce dall'editor memorizzando il file ottenuto
• Stop: esce dall'editor senza memorizzare la variazioni

Caratteristica comune ad ambedue gli editor è l'utilizzo della directory "t" per memorizzare i propri buffer di lavoro.

Come è Amiga Basic ?

Tutti gli appassionati possessori di home o personal computer iniziano la loro carriera di programmatori con lo studio dell'interprete Basic.

Considerato che molti dei possessori di Amiga hanno probabilmente avuto per le mani il glorioso predecessore C/64, ci si interrogherà certamente sulle principali differenze dal linguaggio di programmazione di un computer sofisticato ed evoluto come Amiga.

Si tratta di un Basic avanzato piuttosto standard, molto simile allo MS-Basic del Macintosh ed allo IBM-PC Basic, dotato di estensioni idonee a gestire le possibilità sonore e grafiche esclusive di Amiga.

Amiga Basic deve esser caricato da disco, occupa circa 80K di memoria centrale ed incorpora un (lentissimo) editor "integrale", ovvero operante su tutto lo schermo.

E' comunque possibile usare, come testo sorgente, quello prodotto da un word-processor.

Le caratteristiche che fanno di Amiga Basic una delle versioni più evolute del linguaggio sono:

• *totale assenza dei numeri di linea*
• *blocchi condizionali If.. Then.. Else multipli*
• *sottoprogrammi con variabili allocate dinamicamente*
• *numeri interi a 16-32 bit*
• *numeri in virgola mobile a 32 e 64 bit*
• *etichette (label) alfanumeriche per identificare sezioni del programma (invece dei numeri di linea)*
• *accesso sequenziale e casuale ai file*
• *supporto di periferiche di I/O indipendente per RS-232*
• *porte parallele in grado, grazie al multitasking, di lavorare contemporaneamente all'interprete Basic.*

Ci sembra importante sottolineare che l'allocazione dinamica delle variabili, ovvero la possibilità di eliminare automaticamente dalla memo-

ria alcune variabili non più necessarie all'uscita da un sottoprogramma, consente la ricorsività, caratteristica presente solo in linguaggi più sofisticati (Pascal, Lisp e simili).

Dicevamo che vi sono estensioni del Basic che supportano le caratteristiche esclusive dell'hardware di Amiga; non dimentichiamo i chip VLSI specializzati per la grafica (Agnes) ed il suono (Daphne).

Il Basic prevede:

• generazioni sonore e musicali su quattro voci sincronizzate tramite le specifiche SOUND e WAVE. La prima permette di emettere un suono di frequenza, durata e volume specificati; la frequenza può essere anche una voce tra quattro definite dal programmatore tramite WAVE, che permette di creare voci di qualsivoglia complessità, ottenendo così dal classico "BEEP" dello Spectrum alla sintesi di un quartetto d'archi, il tutto operante in multitasking, ovvero contemporaneamente all'esecuzione del normale programma Basic.

• Sintesi vocale tramite specifiche SAY e TRANSLATE$ operanti con fonemi ed in grado quindi di simulare qualsiasi idioma. Stiamo attendendo il modulo di libreria (un file) che consenta di ottenere automaticamente la lingua italiana oltre a quella americana (standard).

• Capacità di memorizzare, e visualizzare, porzioni dello schermo, anche in alta risoluzione, tramite specifiche GET e PUT.

• Presenza di tutte le istruzioni grafiche più evolute con numerosi parametri specificabili: Line, Circle, Paint (strepitosamente veloce anche per grandi superfici), Area, Areafill e Scroll.

• Gestione di animazioni semplificata tramite specifica "Object", un editor di sprite (figure grafiche) incorporato, e funzione "Collision" per rilevazione di contatti tra figure in movimento.

• Interfacciamento semplificato tramite specifiche Declare e Library con routine scritte in linguaggio Assembler, C, BCPL, Fortran, Pascal, Forth (tutti già disponibili)...

• Gestione completa di schermi multipli e finestre spostabili, ridimensionabili, disattivabili eccetera tramite specifiche Screen e Window.

• Menù di tipo "a sipario" o "pull-down" tramite specifica Menu.

Il manuale, fornito con l'interprete Amiga Basic insieme alla macchina, è realizzato piuttosto bene, senza errori e con esempi chiari. Per darne una descrizione veramente esauriente servirebbe un'enciclopedia: pertanto, come sempre accade tra gli hobbisti, la migliore scuola sarà la pratica, il migliore testo in circolazione l'Amiga stesso ed i migliori maestri la pazienza e la passione, oltre, ovviamente, alla nostra Rivista che, non per niente, è realizzata di veri appassionati di questi mostriciattoli al silicio.

Andare in giro con pochi spiccioli

Dove andremo a finire passeggiando a caso per la città; ed altri momenti casual...

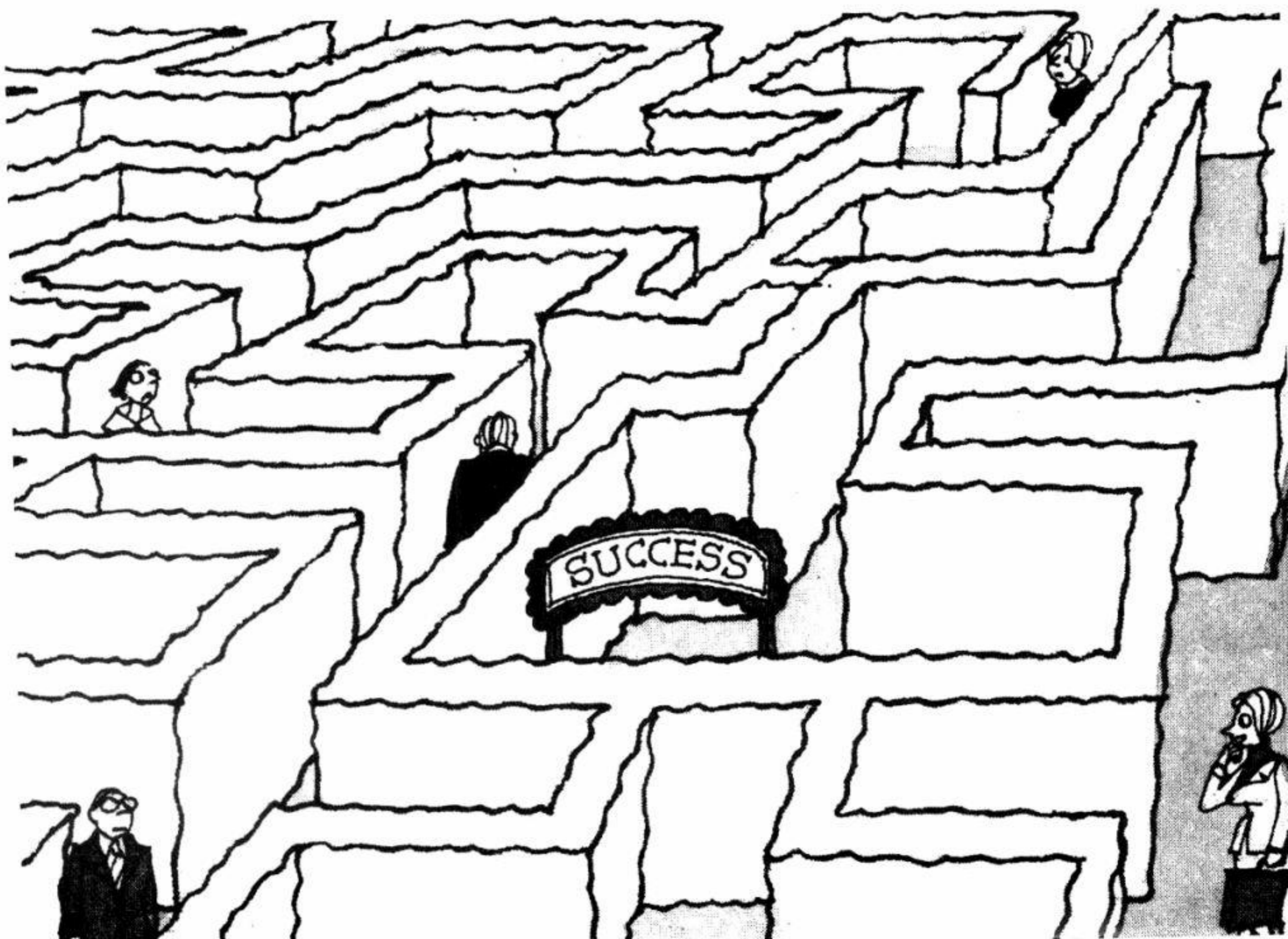
di Valentino Spataro

In questo articolo ci divertiremo con due programmi basati su un fondamentali argomento della statistica.

Il primo simulerà il percorso di una ipotetica passeggiata tra i quartieri della vostra città e risulterà utile per comprendere anche fenomeni fisici.

Il secondo consente la previsione dell'andamento di un fenomeno.

I programmi pubblicati saranno preziosi, tra l'altro, per saperne di più sugli sprite, sulla gestione della bassa o alta risoluzione con la possibilità, nel secondo caso, di mettere a confronto le differenze esistenti tra le routine di Toma e il simulatore Gw-Basic.

Tenete d'occhio anche la particolare gestione dell'output e dell'input perchè non sono state utilizzate le tecniche adoperate di solito.

Hard & soft
LA NIWA
PUÒ ESSERE
LA TUA
MIGLIORE AMIGA
COMMODORE

Una passeggiata

Dopo le applicazioni, apparse nel numero scorso, sulla fondamentale legge della statistica (che indica come calcolare la probabilità che un evento accada), prenderemo stavolta in considerazione un fenomeno più complesso e ne approfitteremo per trarre conclusioni utili per spiegarci fenomeni più complessi.

L'importanza di approfondire gli argomenti proposti trascende il campo statistico: intuirete, infatti, come il metodo usato per spiegarci il fenomeno preso in esame è posto alla base stessa della programmazione, vale a dire rendere semplici problemi apparentemente complessi.

Per seguire il "caso" di cui parleremo tra breve, si consiglia di copiare il programma "Applicazioni", di farlo partire e di scegliere l'opzione 1.

Se avete digitato correttamente il listato sul vostro C/64, compariranno 49 quadrati disposti a rete (7x7), separati tra loro, ed uno sprite quadrato che passa tra gli spazi liberi percorrendo tracciati sempre diversi.

Immaginate di essere lo sprite e che vi muoviate per i quartieri della vostra città. Fate inoltre conto di essere appena usciti da scuola (quadrato nell'angolo in alto a destra) dopo una giornata totalmente negativa, e di aver raggiunto un livello di depressione tale che ad ogni incrocio lanciate una monetina per decidere se dirigervi verso ovest o verso sud (tenete presente che il nord è la parte alta dello schermo); la domanda che sicuramente vi porrete sarà: "Quante probabilità ho di arrivare a casa?" (questa è rappresentata dal quadrato posto in basso a sinistra)

Mentre il programma prosegue all'infinito fino a quando non premerete il tasto F (se volete stampare i dati visualizzati basterà premere il tasto P, dopo aver acceso la stampante) noterete l'aggiornamento continuo dei dati numerici posti verticalmente e orizzontalmente in corrispondenza degli sbocchi Sud e Ovest della periferia: questi indicano il numero di volte in cui voi (=lo sprite) siete passati per quel punto.

Così facendo ottenete quelli che sono detti i risultati "empirici" la cui validità aumenta con l'aumentare del numero delle "passeggiate". Si sottolinea il fatto che, quando l'elaborazione fornisce dati di due cifre, si verificano alcuni inevitabili "slittamenti" che provocano un disallineamento tra il dato visualizzato e l'incrocio cui si riferisce.

Vedremo ora di calcolare, per via matematica, la probabilità di pervenire casualmente a ciascun incrocio.

Premete F per tornare al menù e scegliete l'opzione 2: comparirà una griglia di valori di cui molti sono posti a zero; questi, in seguito, vengono sostituiti con i reali risultati che rappresentano la probabilità di arrivare ad ogni singolo incrocio.

L'unico elemento certo che abbiamo a disposizione è il punto di partenza (la scuola, quadrato in alto a destra): c'è quindi una probabilità su una possibilità (cioè la certezza: 1) di ritrovarci nell'incrocio adiacente alla scuola. Proseguendo nella passeggiata, gli incroci dove possiamo giungere sono due: quello a sud oppure quello a ovest e, di conseguenza, avranno la metà delle probabilità dell'incrocio di provenienza, vale a dire 0.5. Ammettiamo che la monetina che lanciamo ci spinga sempre a ovest: in tutti i successivi incroci avremo sempre la metà delle probabilità di quello di provenienza (1, 0.5, 0.25, 0.125, 0.062).

Si segue lo stesso ragionamento nel caso in cui la moneta indichi sempre il sud. Più problematico, invece, è determinare le probabilità di un incrocio che non sia sui lati estremi della nostra città, ma al suo interno.

Consideriamo l'incrocio a cui possiamo pervenire se la moneta ci dice di andare una volta a ovest e una a sud (oppure al contrario, tanto vi si arriva ugualmente). Possiamo giungervi provenendo da un incrocio posto a nord oppure da uno posto a est. Entrambi vanno cansiderati nel conteggio delle probabilità dell'incrocio in esame. In particolare, abbiamo metà probabilità per l'incrocio a nord e metà per quello a est.

La somma di queste probabilità fornisce quella di arrivare all'incrocio considerato. Se, ad esempio, quello a nord ha 0.5 probabilità e quello a est 0.5, l'incrocio comune cui possiamo pervenire ha la probabilità:
0.5/2 + 0.5/2 = 0.5

Continuando lo stesso ragionamento per gli altri incroci, arriviamo a determinare le probabilità di arrivare all'incrocio di casa nostra (quadrato in basso a sinistra).

Confrontate ora i risultati empirici (opzione 1, dopo almeno un centinaio di passeggiate) con quelli teorici: se avete la stessa fortuna di Gastone risulteranno uguali; se invece siete Paperino, provate e riprovate: dopo centinaia di tentativi dovreste ottenere risultati quasi corrispondenti. In tutti gli altri casi i risultati possono esser considerati simili per la convalida del calcolo.

Con il semplice esempio descritto avrete certamente notato come la probabilità di arrivare a casa dipenda in maniera determinante dalla probabilità degli incroci precedenti.

Possiamo affermare, più in generale, che per calcolare le probabilità che si realizzi una certa situazione all'interno di un fenomeno complesso, bisogna prima calcolare tutte le probabilità dei singoli casi che portano alla situazione in esame.

Con l'opzione 2, infatti, vengono calcolate prima le probabilità dei casi precedenti: solo procedendo in questo modo possiamo calcolare le probabilità di arrivare a casa in tempo per l'ora di pranzo.

Nei casi più complessi, comunque, si preferisce calcolare empiricamente subito quante volte si realizza un fenomeno e in base ai risultati dire quante probabilià ha di realizzarsi in futuro tenendo come valida la legge dei grandi numeri (vedi CCC N.42) secondo la quale quanto più si ripete l'esperimento tanto più ci si avvicina ai risultati teorici.

Le considerazioni finora sostenute, vengono applicate anche nella Fisica con rilevante influenza.

Infatti, oggi come oggi, la fisica classica non ha più il carattere di assoluta certezza di un tempo, ma probabilistico; non si ritiene più impossibile che un pianeta vada contro la legge della gravitazione universale di Newton, ma si ritiene questo fatto altamente improbabile.

Directory

Tutti i programmi pubblicati su questo numero di Commodore Computer Club, sono registrati su un dischetto appartenente alla serie "Directory".

Oltre ai programmi citati, sono presenti altri file di notevole interesse per coloro che desiderano realmente utilizzare il proprio computer.

Sono infatti presenti, di solito, anche i programmi pubblicati sull'altra nostra rivista "Personal Computer", ed altri file che, in totale, riempono quasi per intero i 664 blocchi normalmente disponibili su un floppy disk.

Sono disponibili i seguenti dischetti:

***Directory** N. 1 (CCC N.34 + raccolta dell'intera Enciclopedia di routine)*
***Directory** N. 2 (CCC N.35 + CCC N.36)*
***Directory** N. 3 (CCC N.37 + PC N.7 + PC N.8)*
***Directory** N. 4 (CCC N.38 + file vari)*
***Directory** N. 5 (CCC N.39 + PC N.9)*
***Directory** N. 6 (CCC N.40 + PC N.10)*
***Directory** N. 7 (CCC N.41 + file vari)*
***Directory** N. 8 (CCC N.42 + file vari)*
***Directory** N. 9 (CCC N.43 + file vari)*

Come procurarsi i dischetti della serie "Directory"

Avvertiamo i lettori che NON è assolutamente possibile inviare i programmi su nastro, per intuibili motivi di economia ed affidabilità del nastro cassetta.

Ogni numero di "Directory" può quindi esser richiesto SOLO su disco inviando L.12000 per ciascun disco oltre a L.3000 (fisse) per le spese di imballo e spedizione (indipendenti dal numero di dischi richiesti).

Chi desiderasse la spedizione raccomandata, deve aggiungere altre 3000 lire per l'ulteriore affrancatura.

Non ci è possibile inviare materiale contrassegno: si prega di astenersi dal chiedere eccezioni alla regola.

Compilate un normale modulo di C/C postale indirizzando a:

C/C postale N. 37952207
Systems Editoriale
Viale Famagosta, 75
20142 Milano

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento") non solo il vostro nominativo completo, ma anche il numero del disco desiderato; esempio:

"Directory N.1"
"Directory N.3"
"Directory N.4"

Totale:
L.12000x3 +
L.6000 (spediz. racc.)
= L.42000

(spese di imballo e spediz. racc. comprese).

N.B. Per ottenere il materiale ordinato in tempi più ristretti, inviate l'importo a mezzo assegno bancario non trasferibile con lettera di accompagnamento: le poste italiane non brillano per velocità! (due mesi circa per il recapito di un C/C postale).

E ancora: perchè tutte le molecole dell'aria di una stanza non si spostano tutte in una sola parte di essa? Questo fenomeno è possibile, ma fortunatamente improbabile: le probabilità sono le stesse di ottenere tutte "teste" lanciando tante monetine quante sono le molecole dell'aria.

Nel primo programma avrete notato che è presente un'altra opzione di cui ora vedrete l'utilità.

Tartaglia e le monete

Studiando la matematica vi sarete certamente imbattuti nel famigerato triangolo di Tartaglia (detto anche di Pascal); è un metodo semplice per calcolare i valori da attribuire ai termini noti nello sviluppo di un binomio di ennesimo grado come, ad esempio:

$(a+b) \uparrow n$

Tale espressione matematica è spesso ricorrente nella statistica: si è infatti notato che, lanciando M monete N volte, il numero di teste e di croci che ne risultano portano a risultati del tutto simili a quelli forniti dal triangolo di Tartaglia; vedremo, quindi, di calcolare la probabilità che si presenti una qualunque delle varie combinazioni di testa e croce, che possono risultare dal lancio di 2, 3, oppure 4 monete. Le tre relazioni matematiche necessarie sono le seguenti:

$(c+t) \uparrow 2 = c \uparrow 2 + 2*c*t + t \uparrow 2$

$(c+t) \uparrow 3 = c \uparrow 3 + 3*c \uparrow 2*t + 3*c*t \uparrow 2 + t \uparrow 3$

$(c+t) \uparrow 4 = c \uparrow 4 + 4*c \uparrow 3*t + 6*c \uparrow 2*t \uparrow 2 + 4*c*t \uparrow 3 + t \uparrow 4$

in cui, ovviamente, "t" significa testa e "c" croce. Interpretiamo i dati riportati:

L'esponente associato al termine (c+t), al primo membro di ciascuna equazione, rappresenta il numero delle monete lanciate; ne consegue che la prima equazione si riferisce al lacio di due monete (esponente=2); la seconda considera tre monete (esponente=3) eccetera.

Gli esponenti che troviamo associati agli addendi del secondo membro di ciascuna equazione, rappresentano il numero di teste e di croci. Per esempio...

$c \uparrow 3*t \uparrow 2$

...rappresenta tre croci e due teste.

Il coefficiente numerico in ciascuna espressione al secondo membro dice in quanti modi può presentarsi ciascun evento.

Dunque, se si lanciano tre monete (vedi seconda equazione), sappiamo che tre croci si presentano in un solo modo (c 3), due croci e una testa in tre modi (3*c 2*t), e tre teste in un solo modo (t 3). Sommando i vari coefficienti (ci riferiamo ancora alla seconda equazione) si ottiene 1+3+3+1=8, che è il numero degli eventi possibili, cioè il numero totale di possibilità che possano verificarsi.

Nel N.42 di CCC ci siamo riferiti ad una formula basilare, che ora richiamiamo brevemente:

$Npr=Nfa/Npo$

Npr= n. probabilità
Nfa= n. casi favorevoli
Npo= n. casi possibili

La probabilità, dunque, che si presentino due croci e una testa è:

$P(c \uparrow 2*t \uparrow 1)=3/8$ *(cioè: 37.5%)*

in cui P=probabilità che si verifichi il fenomeno indicato tra parentesi.

Questo tipo di analisi è valida solo se la probabilità di ciascuno dei singoli risultati è esattamente 1/2 (come quando si lancia una moneta non truccata; anzi provate voi a truccarla seguendo i suggerimenti dell'inserto del N.38 di CCC).

Per vedere il triangolo di Tartaglia vi basta usare il programma "Applicazioni": se userete potenze inferiori al 9 vedrete il triangolo, altrimenti solo il risultato finale.

Per lanciare le monete non prendete il salvadanaio, ma copiate il programma nella versione Gw-Basic oppure Toma (purchè contenga il comando Char) che forniscono gli stessi risultati; in questo modo potrete anche confrontare i due linguaggi.

Fate girare il programma e digitate un numero di monete compreso tra 1 e 9 (estremi compresi). Potete impostare il numero di lanci che volete: il valore consigliato, per non aspettare molto e avere risultati apprezzabili, è 100.

Dopo una breve pausa comparirà una schermata in cui al di sotto del grafico resta uno spazio diviso in tre parti. Nella prima di queste è indicato il numero di teste e di croci relative a ogni colonna (la loro somma fornisce, ovviamente, il numero di monete selezionato). Nella parte centrale sono riportati i risultati teorici calcolati con Tartaglia (cioè i valori numerici della "base" del triangolo e le percentuali corrispondenti).

Per sapere quante volte esce una testa e 8 croci lanciando 9 monete 100 volte, basterà guardare i risultati posti sotto la seconda colonna di sinistra. Nella parte a metà è presente un 9 (che deriva dal triangolo elaborato con il programma precedente) e la percentuale dell'1 per cento. Nella parte sottostante, invece, troverete i risultati ottenuti al termine del vostro esperimento.

Per motivi di spazio, ovviamente, sono riportati solo le parti intere delle percentuali calcolate. La presenza di eventuali 0% indica, pertanto, che la probabilità calcolata è inferiore all'1% e non che... non esiste!

SCHEDA TECNICA

Software didattico per applicazioni di statistica.

Idoneo per computer C/64, ma adattabile, pur se con qualche difficoltà, ad altri computer Commodore.

Due dei listati che compaiono richiedono il preventivo caricamento delle routine grafiche di Toma oppure del simulatore di GW-Basic

Anche i programmi pubblicati in queste pagine sono contenuti nel disco "Directory" di questo mese.

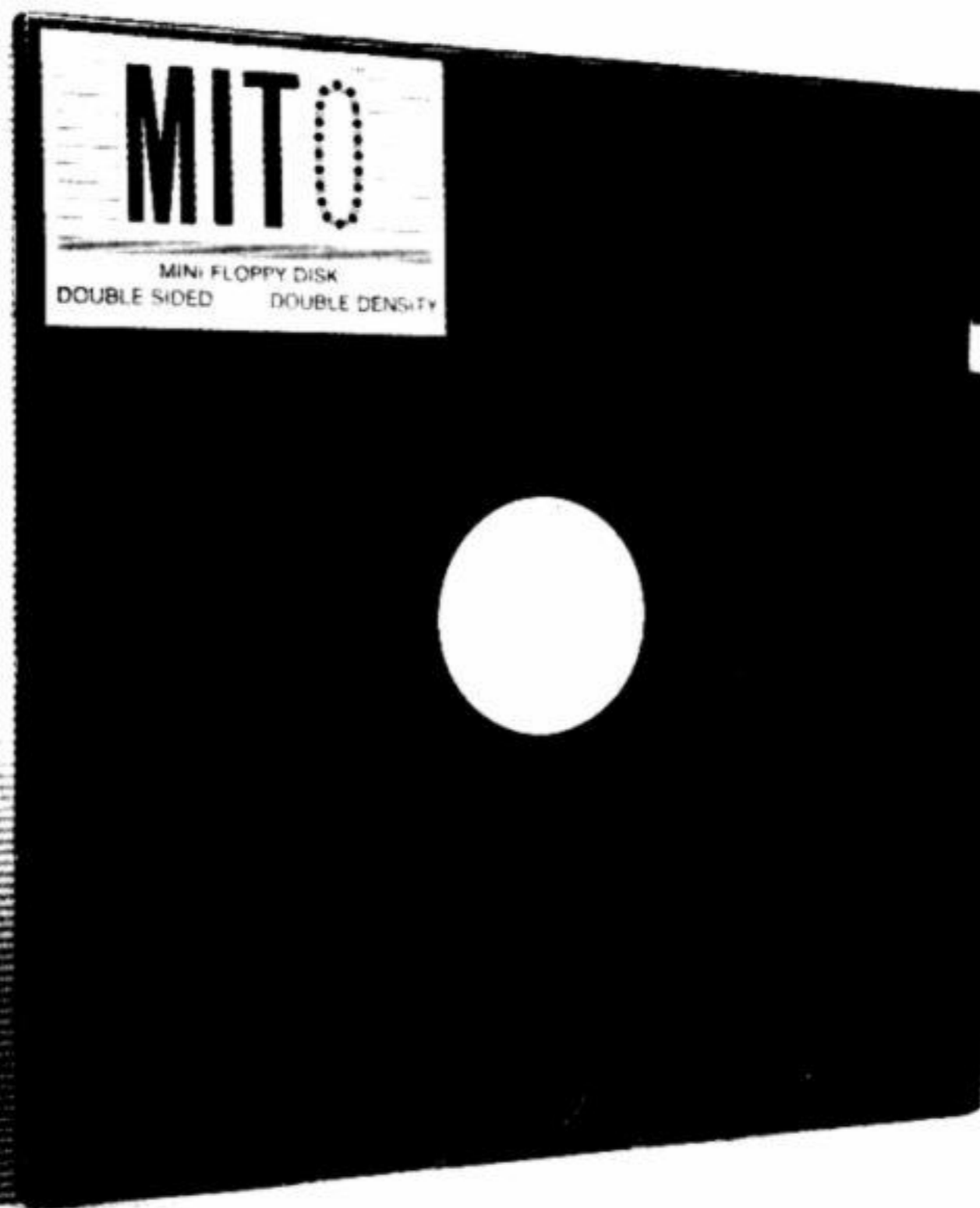
```
110 REM APPLICAZIONI DI
120 REM STATISTICA PER C-64
130 REM BY VALENTINO SPATARO
140 :
150 DIM  A(80):POKE 53269,0:REM
     SPEGNE SPRITE
160 PRINTCHR$(147):PRINT "MENU'
170 PRINT"1 PASSEGGIATA A CASO:
     P=STAMPA DATI
                        F=FINE"
180 PRINT"2 CALCOLO PROBABILITA
    ' DELLA PASSEGGIATA"
190 NN=1:REM N.PASSEGGIATE
200 PRINT"3 TARTAGLIA":PRINT:PR
    INT"0 FINE
210 GET A$:IF A$="0" THEN END
220 ON  VAL(A$) GOTO 240,590,75
    0
230 GOTO 210
240 REM PREPARA LO SCHERMO
250 A$=" ┌─┐":B$=" └─┘":E$=" |
     |":REM  CHR$=32,176,99,174/
    32,173,99,189/32,125
260 FOR A=1 TO 7:C$=C$+A$:NEXT
270 FOR A=1 TO 7:D$=D$+B$:NEXT
280 FOR A=1 TO 7:F$=F$+E$:NEXT:
    PRINTCHR$(147)
290 FOR A=1 TO 7:PRINTC$:PRINTF
    $:PRINTD$:NEXT
300 REM INIZIALIZZA
310 X=205:Y=68:VX=X:VY=Y:SX=32:
    SY=24:RT=1
320 V=53248:POKE V+21,4:POKE 20
    42,13:POKE V+41,1
330 POKE V+4,X:POKE V+5,Y
340 REM DISEGNA SPRITE
350 FOR A=0 TO 62:POKE 832+A,25
    5:NEXT
360 FOR A=0 TO 62 STEP 3:POKE 8
    32+A,0:NEXT
370 FOR A=0 TO 21:POKE 832+A,0:
    NEXT
380 REM ROUTINE PRINCIPALE
390 A=INT(RND(1)*2)+1:GOSUB 530
400 IF A=1 THEN X=X-SX:FOR A=VX
     TO X STEP -1:POKE V+4,A:NE
    XT
410 IF A=2 THEN Y=Y+SY:FOR A=VY
     TO Y:POKE V+5,A:NEXT
420 IF X<SX OR Y>=213 THEN GOSU
    B 460:X=205:Y=68:POKE V+4,X
    :POKE V+5,Y
430 VX=X:VY=Y
440 GOTO 390:REM FINE ROUTINE P
    RINCIPALE
450 REM MEMORIZZA IN MATRICI DO
    VE E' ARRIVATO LO SPRITE
460 IF Y<212GOTO 480
470 A=(205-X)/32+1:A(A)=A(A)+1:
    GOTO 500
480 A=(212-Y)/24+8:A(A)=A(A)+1
490 REM STAMPA SU SCHERMO IL CO
    NTENUTO DELLE MATRICI
500 PRINT"[HOME]":FOR A=14 TO 9
     STEP -1:PRINT:PRINT:PRINTA
    (A):NEXT:PRINT:PRINT:PRINT
505 FOR A=7 TO 1 STEP -1:PRINTA
    (A)"  ";:NEXT
510 PRINT"[HOME]N.PASSEGGIATE E
    FFETTUATE: "NN:NN=NN+1:RETU
    RN
520 REM STAMPA SU PRINTER
530 IF PEEK(198)=0 THEN RETURN
540 GET A$:IF A$="F" THEN GOSUB
     910: RUN
550 IF A$<>"P" THEN RETURN
560 A$="":OPEN 4,4:FOR A=14 TO
    1 STEP -1:IF A>8 THEN PRINT
    #4,A(A)
570 IF A<8 THEN PRINT#4,A(A);
580 NEXT:PRINT#4:CLOSE 4:RETURN
590 PRINTCHR$(147):REM INIZIALI
    ZZA
600 A(7)=1:FOR A=1 TO 6
610 A(7-A)=1/2↑A:A((A+1)*7)=1/2
    ↑A
620 NEXT:GOSUB 680
```

```
630 REM CALCOLA PROBAB. E METTE
     IN SCHEMA
640 FOR  X1=1 TO 6
650 FOR  X2=6  TO  1  STEP -1:W
    =X1*7+X2
660 A(W)=A(W-7)/2+A(W+1)/2:X=X2
    *5:Y=X1*2+1:GOSUB  920
670 NEXT:NEXT:GOSUB  910: RUN
680 REM VISUALIZZA
690 PRINTCHR$(19):FOR  A=0  TO
     6
700 FOR  B=1  TO  7:Y=A*2+1:X=B
    *5
710 W=A*7+B:GOSUB 920
720 NEXT:
730 NEXT:RETURN
740 GOSUB 910: RUN
750 PRINTCHR$(147):INPUT "ESPON
    ENTE ";P:P=P-1:IF P=-1 THEN
      RUN
760 DIM C(P+1),B(P+1)
770 C(0)=1:C(1)=1:X=1:B(0)=1:B(
    1)=1:GOTO 820
780 IF X>P THEN F$="I":GOSUB 83
    0:GOSUB 910: RUN
790 FOR A=1 TO X
800 B(A)=C(A)+C(A-1)
810 NEXT:B(A)=1:X=X+1
820 IF P>7GOTO 870
830 Y=19-(2*X):FOR A=0 TO X
840 IF F$<>"I" THEN PRINT TAB(Y
    ) B(A);:Y=Y+4:GOTO 860
850 POKE 199,1:PRINTB(A);
860 NEXT:PRINT:IF F$="I" THEN R
    ETURN
870 FOR A=1 TO X
880 C(A)=B(A)
890 NEXT
900 GOTO 780
910 PRINT"PREMI UN TASTO":POKE
    198,0:WAIT 198,1:RETURN
920 POKE 211,X:POKE 214,Y:SYS58
    640:REM  FUNZIONE PRINT AT
    O LOCATE
930 A$=STR$(A(W)):X=LEN(A$):IF
    X>5 THEN X=5
940 A$=LEFT$(A$,X):PRINTA$:RETU
    RN
950 END
```

```
10 REM STATISTICA: APPLICAZIZO
   NI
20 REM LANCIO DI MONETE
30 REM E' INDISPENSABILE CARIC
   ARE L'ULTIMA VERSIONE DELLE
     ROUTINE DI TOMA
40 REM BY VALENTINO SPATARO
50 REM:
60 PRINTCHR$(147)
70 INPUT"NUMERO MONETE";P:P=P-
   1:IFP=-1ORP>8THENRUN
80 INPUT"NUMERO LANCI";LA:IFLA
   =0GOTO80
90 DIMA(P+1),B(P+1)
100 A(0)=1:A(1)=1:X=1:B(0)=1:B(
    1)=1
110 IFX>PTHEN170
120 REM CREA IL TRIANGOLO DI TA
    RTAGLIA E CONSIDERA SOLO L'
    ULTIMO
130 FORA=1TOX
140 B(A)=A(A)+A(A-1)
150 NEXT:B(A)=1:X=X+1
160 FORA=1TOX:A(A)=B(A):NEXT:GO
    TO110
170 POKE50151,75:DY=100-PEEK(50
    151):REM CAMBIA PUNTO DI VI
    STA
180 REMINIZIALIZZA TUTTO
190 ←CLEAR:←GRAF 0,7:←COLOR 1:P
    =P+1
200 FORA=-160TO160STEP 320/(P+1
    )
210 ←DRAW A,0,Z,A,100+DY,Z:REM
    DIVIDE IN COLONNE
220 NEXT:←DRAW -160,0,Z,160,0,Z
230 REM VISUALIZZA TUTTI I DATI
     (V. ARTICOLO)
240 FORA=0TOP:←CHAR A*40/(P+1),
    19,0,B(A):TT=TT+B(A):NEXT:T
    T=100/TT
250 FORA=P TO 0 STEP-1:←CHAR A*
    40/(P+1),20,0,STR$(INT(B(A)
    *TT))+"%":NEXT
260 FORA=0TOP:←CHAR A*40/(P+1),
    16,0,STR$(A)+"T"
270 ←CHAR A*40/(P+1),17,0,STR$(
    P-A)+"C":NEXT:
280 FORA=0TOP:B(A)=0:NEXT:REM I
```

```
    NIZIALIZZA
290 ←DRAW -160,-22,Z,160,-22,Z:
    ←DRAW $,-45,$,$,-45,$
300 REM **** INIZIA LANCI ****
310 SY=125/(LA/2):FORA=1TOLA:T=
    0
320 REM LANCIA P MONETE
330 FORB=1TOP:GOSUB410:A(B)=RS:
    IF RS=1 THEN T=T+1:REM CONT
    A LE TESTE
340 NEXT:
350 B(T)=B(T)+1
360 ←DRAW T*320/(P+1)-160+5 ,B(
    T)*SY,Z,T*320/(P+1)-160+320
    /(P+1)-5,B(T)*SY,Z:NEXT
370 REM STAMPA I RISULTATI FINA
    LI
380 FORA=0TOP:←CHAR A*40/(P+1),
    22,0,B(A):NEXT
390 LA=100/LA:FORA=P TO 0 STEP-
    1:←CHAR A*40/(P+1),23,0,STR
    $(INT(B(A)*LA))+"%":NEXT
400 WAIT198,1:END
410 RS=INT((RND(1)*2)+1):RETURN
    :REM ESTRAZIONI
420 END
```

```
100 REM STATISTICA CON IL C/64
110 REM LANCIO DI N MONETE
120 REM RICHIEDE SIMULATORE DI GW-BASIC
130 REM BY VALENTINO SPATARO
140 :
150 PRINTCHR$(147)
160 INPUT"NUMERO MONETE";P:P=P-1:IFP=-1ORP>8THENRUN
170 INPUT"NUMERO LANCI";LA:IF LA=0 GOTO170
180 DIMA(P+1),B(P+1)
190 A(0)=1:A(1)=1:X=1:B(0)=1:B(1)=1
200 IFX>PTHEN260
210 REM CREA IL TRIANGOLO DI TARTAGLIA E CONSIDERA SOLO L'ULTIMO
220 FORA=1TOX
230 B(A)=A(A)+A(A-1)
240 NEXT:B(A)=1:X=X+1
250 FORA=1TOX:A(A)=B(A):NEXT:GOTO200
260 :
270 REM ***** INIZIALIZZA HI-RES *****
280 CLS 1:SCREEN 1:COLOR 15,0,0:P=P+1
290 FORA=0TO320STEP 320/(P+1)
300 LINE (A,0)-(A,125):REM DIVIDE IN COLONNE
310 NEXT:LINE (0,125)-(320,125)
320 REM DATI TEORICI (V. ARTICOLO)
330 FORA=0TOP:LOCATE (320*A/(P+1),152):PRINTB(A):TT=TT+B(A):NEXT:TT=100/TT
340 FORA=P TO 0 STEP-1:LOCATE (320*A/(P+1),160):PRINTSTR$(INT(B(A)*TT))+"%":NEXT
345 REM LEGENDA PER IL GRAFICO
350 FORA=0TOP:LOCATE (A*320/(P+1),16*8):PRINTSTR$(A)+"T"
360 LOCATE (A*320/(P+1),17*8):PRINTSTR$(P-A)+"C":NEXT:
370 FORA=0TOP:B(A)=0:NEXT:REM 'VUOTA' B(A)
380 LINE (0,173)-(320,173):LINE (0,195)-(320,195)
390 LINE (0,149)-(320,149):LINE (0,195)-(320,195)
400 REM **** INIZIA LANCI ****
410 SY=125/(LA/2):FORA=1TOLA:T=0
420 REM LANCIA P MONETE
430 FORB=1TOP:GOSUB510:A(B)=RS:IF RS=1 THEN T=T+1:REM CONTA LE TESTE
440 NEXT:
450 B(T)=B(T)+1
460 LINE (T*320/(P+1)+5,125-B(T)*SY)-(T*320/(P+1)+320/(P+1)-5,125-B(T)*SY):NEXT
470 REM VISUALIZZA I RISULTATI OTTENUTI EMPIRICAMENTE
480 FORA=0TOP:LOCATE (A*40/(P+1)*8,22*8):PRINTB(A):NEXT
490 LA=100/LA:FORA=P TO 0 STEP-1:LOCATE (A*40/(P+1)*8,23*8)
500 PRINT STR$(INT(B(A)*LA))+"%":NEXT:WAIT198,1:SCREEN 0:RUN
510 RS=INT((RND(1)*2)+1):RETURN:REM ESTRAZIONI
520 END

READY.
```

LA PERFEZIONE DIVENTA MITO

MITO - 5 1/4" Floppy 48 TPI
Doppia Faccia - Doppia Densità
Garantito al 100% - Velocità di registrazione 5800 BPI
600.000 bytes unformatted.

le misure della perfezione

RECOVERY SERVICE - Un nostro servizio esclusivo. Cosa è il Recovery Service? È uno scudo a protezione del vostro lavoro. Se per un incidente qualsiasi: macchie di caffè, di cioccolato o impronte, il vostro disk dovesse danneggiarsi, la MICROFORUM è in grado di recuperare i dati senza alcun esborso da parte vostra.

La *MICROFORUM MANUFACTURING INC.* è interessata all'ampliamento della propria rete distributiva. Per qualsiasi contatto scrivere anche in italiano.

944/A St. Claire Ave. West TORONTO, CANADA M6C 1C8 Tel. (416) 656-6406 - Tlx. 06-23303 MICROFORUM TOR Telefax (416) 656-6368

STUDIO C SANTORI Roma

Dimmi un numero!

Come "passare" parametri dal Basic al Linguaggio Macchina; e altre storie sulle Rom del C/64

di Giancarlo Mariani

Molti lettori hanno spesso digitato brevi routine in Linguaggio Macchina (LM) ma, pur soddisfatti del loro funzionamento, desiderano sapere in che modo agiscono.

In questo articolo spiegheremo nei minimi dettagli come agire per realizzare una routine LM da richiamare a piacimento in nostri programmi Basic.

Che cos'è un parametro?

Molte routine, per funzionare, non necessitano di alcun parametro, ossia basta richiamarle con "SYS ind" (in cui "ind" è l'indirizzo di partenza della routine) per far compiere il lavoro richiesto.

Il più delle volte, comunque, le routine sviluppate per funzionare necessitano di parametri, che devono essere "passati" al momento della chiamata.

Un parametro è un valore (numerico o stringa), sul quale la routine si basa per svolgere la funzione. Esaminiamo qualche esempio:

- *una routine che esegua la somma di due numeri avrà bisogno di due parametri esterni, che sono appunto i due numeri da sommare tra loro.*
- *una routine di logaritmo necessiterà di un solo parametro (l'argomento del logaritmo).*
- *una routine di cancellazione schermo non necessita di alcun parametro.*

I parametri, se ci sono, possono essere indicati dall'"esterno", ossia tramite un programma Basic. Esistono due strade che un programmatore

può seguire per raggiungere il suo scopo, e che esaminiamo ora nei dettagli.

I parametri nelle locazioni

E' il sistema più semplice, ma anche il meno pratico dal momento che consiste nell'utilizzare alcune locazioni di memoria come "parcheggio" per un successivo scambio di dati tra Basic e LM.

Supponiamo di dover scrivere una routine che serva ad alterare i colori di sfondo e di bordo del video; necessita, ovviamente, dei due parametri "esterni" (i due colori).

Potremo, quindi, POKare i due valori in opportune celle di memoria (che chiamiamo LOC1 e LOC2) e, in seguito, richiamare la routine LM che provvederà a leggere le due celle ed a trasferirne il contenuto negli appositi registri di colore.

La routine mista Basic-LM (appena accennata) può essere di questo tipo:

```
10 rem programma Basic
15 print"digita colore sfondo"
20 input a
25 print"digita colore bordo"
30 input b
35 rem salva i due valori in loc1 E
loc2
40 poke loc1,a
45 poke loc2,b
50 rem chiama la routine LM
55 sys ind:rem ind=indirizzo partenza
```

La routine LM, chiamata "ind", sarà del tipo:

```
LDA LOC1  ;Carica in A col. sfondo
STA 53281 ;Cambia sfondo
LDA LOC2  ;Carica in A col. bordo
STA 53280 ;Cambia bordo
RTS       ;Ritorna al basic
```

Il programma è talmente semplice che si commenta da solo. Si può facilmente notare, però, che all'aumentare dei parametri da passare, la stesura del programma risulta macchinosa, il sistema poco pratico ed il programma finale praticamente illeggibile, a causa della massiccia presenza di istruzioni POKE.

Se, poi, i parametri da passare sono numeri molto grandi (o stringhe), una gesitione di questo tipo diventa praticamente impossibile.

Per fortuna "mamma" Commodore ci viene incontro, offrendo un S.O. ricco di routine adatte a semplificare notevolmente il lavoro.

LA PERFEZIONE DIVENTA MITO

QUAD-MITO - 5 1/4" 96 TPI DS/QD
Floppy disk a quadrupla densità, disegnato per aumentare la capacità di registrazione sino a 780 kb per dischetto.
Velocità di registrazione 5800 BPI

MEGA-MITO - 5 1/4" 96 TPI HIGH DENSITY
Floppy ad alta densità, disegnato per drive da 1.2 MEG (AT e compatibili).
Velocità di registrazione 9650 BPI

MICRO-MITO - 3 1/2" 135 TPI DS/DD
Costruito per l'era dei disk drive da 3 1/2".
Velocità di registrazione 8100 BPI

le misure della perfezione

La ***MICROFORUM MANUFACTURING INC.***
e interessata all'ampliamento della propria rete distributiva.
Per qualsiasi contatto scrivere anche in italiano.

944/A St. Claire Ave. West TORONTO, CANADA M6C 1C8
Tel. (416) 656-6406 - Tlx. 06-23303 MICROFORUM TOR
Telefax (416) 656-6368

STUDIO C.SANTORI Roma

Il passaggio automatico dei parametri

Prima di passare alla spiegazione del metodo più efficace, è necessaria una premessa:

Nel C/64 (ma anche in altri Commodore) esistono due particolari locazioni di memoria (nel nostro caso $7A e $7B) che servono per far sapere al S.O. quale carattere è attualmente elaborato.

In seguito, ad esempio, ad un qualsiasi comando (es. Print "pippo") il "puntatore" formato dalle due locazioni $7A e $7B, non appena premete il tasto Return, indirizzerà il primo carattere del comando, ossia la "P", poi il secondo ("R"), il terzo ("I"), e così via fino al termine del comando.

Una volta riconosciuto il comando, dotato di eventuali parametri, questo viene eseguito ed il puntatore resta posizionato sul carattere successivo all'ultimo letto. Per esempio, supponendo di dare un comando del tipo...

Print B:B=B+10

...una volta eseguito il comando Print B, il puntatore resterà posizionato sul carattere doppio punto (:) per permettere al S.O. di leggere e riconoscere anche il successivo comando (B=B+10).

Tale caratteristica può essere sfruttata a nostro vantaggio: dopo aver dato un comando del tipo "SYS ind", abbiamo la certezza che il puntatore dei caratteri sarà posizionato immediatamente dopo l'ultima cifra dell'indirizzo: in questo spazio potremo inserire i valori da passare, come parametri, alla nostra routine LM.

Il lavoro da compiere risulta, in realtà, meno complesso di quanto sembri, grazie al fatto che alcune routine su Rom provvedono a leggere automaticamente il parametro posto dopo un comando, a controllarlo sintatticamente ed a inserirlo in una forma facilmente utilizzabile dal programmatore, provvedendo inoltre ad aggiornare il puntatore del carattere corrente.

Il lavoro per l'utente si limita, quindi, al richiamo di più routine del S.O.

Riprendiamo l'esempio precedente (cambio colore sfondo-bordo) in cui i parametri necessari sono due. Questi andranno, ovviamente, separati tra di loro e da una virgola. La sintassi del comando può essere la seguente:

SYS IND,CS,CB

in cui CS è il colore dello sfondo mentre CB è il colore del bordo. La routine "ind" in LM per assolvere a questa funzione può essere la seguente:

Routine 1

```
JSR $AEFD ;Presenza virgola?
JSR $B79E ;Lettura primo param.
STX 53281 ;Cambio colore sfondo
JSR $AEFD ;Come sopra
JSR $B79E ;Lettura secondo par.
STX 53280 ;Cambio colore bordo
RTS
```

La prima subroutine richiamata (JSR $AEFD) esegue un salto ad una routine del S.O. che provvede a controllare se il carattere corrente (quello indirizzato dal puntatore) è una virgola; in caso contrario verrà, dalla stessa routine, emesso un Syntax error.

E' importante notare che tra le virgole ed i vari parametri possono essere aggiunti spazi vuoti: questi, grazie alla stessa routine, verranno completamente ignorati in fase di lettura del parametro.

JSR $B79E richiama un'altra routine che "prende" il parametro numerico (compreso tra 0 e 255) indirizzato dal puntatore e lo trasferisce nel registro X del microprocessore 6510. Anche in questo caso verrà visualizzato un Syntax error (se il numero non è presente) un Type mismatch (in caso di presenza di una stringa) e un Illegal quantity (se il numero è esterno all'intervallo 0/255).

Resta da attuare il trasferimento del registro X nella locazione del colore dello sfondo. A questo provvede la STX 53281.

Il programma LM continua in modo identico, con la sola eccezione che questa volta il parametro andrà nel registro di colore del bordo (53280) anzichè in quello di sfondo.

Una cosa importantissima da notare è che tutte le routine S.O. viste sinora (e quelle che vedremo in seguito), aggiornano automaticamente il puntatore del carattere all'ultimo letto più uno. Come facilmente intuibile, questo semplifica enormemente il lavoro del programmatore.

Anche se il programmino LM funziona bene così come è pubblicato, possiamo migliorarlo ulteriormente inserendo controlli aggiuntivi al parametro letto.

Noi sappiamo che i colori possibili sono solo 16 (numerati da 0 a 255) ed è inutile accettare tutti i numeri compresi tra 0 e 255 che risulterebbero ripetitivi. E' sicuramente funzionale limitare il "range" dei valori inseribili tra 0 e 15, dando apposito errore in caso contrario.

La routine per limitare l'accettazione del range potrebbe essere:

*Routine 2 **

```
JSR $AEFD  ;Controllo virgola
JSR $B79E  ;Lettura parametro
CPX #15    ;E' magg/uguale 16?
BPL error  ;Si: salta all'errore
........ ;No: continuo
RTS
error:
  LDX #$0E  ;Punta a Illeg.quan.
  JMP($0300) ; Genera errore
```

Effettuato il controllo del parametro (CPX #16), viene effettuato (tramite BPL) un salto se il contenuto del registro X (che è poi il parametro letto, vedi JSR $B79E) è maggiore o uguale a 16.

Nel caso in cui X sia minore di 16 (ossia compreso tra 0 e 15; ricordiamo che la routine a $B79E non accetta valori minori di zero), la routine continua con la gestione del parametro, che varia a seconda del lavoro da svolgere. Se il parametro invece è maggiore di 15, viene chiamata la routine di errore.

La routine di errore è già presente nel S.O. del C/64, e provvede a stampare gli errori tipici del Basic (Syntax, Out of Data, ecc.). Questa routine è indirizzata dalle due locazioni $0300 e $0301, e deve quindi essere richiamata tramite un JMP indiretto a $0300. Nella locazione $300 (dec.768) e nell'adiacente $0301 (769) è contenuto, infatti, l'indirizzo reale della routine di errore:

*Peek(768) + Peek(769)*256=58251*

La "error routine" necessita anche di un parametro in ingresso, che deve essere passato tramite il registro X, e che contiene il codice (o il numero) dell'errore da visualizzare e deve essere compreso tra 1 e 30: valori esterni produrranno la stampa di caratteri casuali.

Per ciò che riguarda la seconda routine, nel caso in cui il parametro sia maggiore di 15, viene richiamata la routine di errore con un codice di ingresso= $0E che riproduce l'Illegal Quantity.

Provate a digitare, servendovi di un monitor di LM, questa brevissima routine:

** Routine 3 **

```
JSR $AEFD   ;Controllo virgola
JSR $B79E   ;Lettura parametro
  e trasf. nel reg. X
JMP ($0300) ;Generaz. errore a
  seconda di X
```

Poi, da Basic, impartite...

SYS ind,err

"Err" è ovviamente il codice dell'errore, compreso tra 1 e 30. Che cosa succede? Ogni volta che si richiama la routine viene generato e stampato un errore, che dipende dal parametro passato. E' da notare che non vi è alcuna differenza tra un errore capitato accidentalmente durante la stesura o la prova di programmi basic, ed un errore generato "artificialmente"; infatti, anche in quest'ultimo caso si produrranno esattamente gli stessi effetti, quali reset di alcuni puntatori, arresto del programma eventualmente in corso, ecc.

Anche in Basic è possibile realizzare quanto appena accennato. Forse non tutti sanno che è possibile, alterando alcune locazioni di memoria, inserire nell'accumulatore, nel registro X, in quello Y (e addirittura nello Stack Pointer), altrettanti valori prima di effettuare una qualunque Sys:

780: Accumulatore
781: Registro X
782: Registro Y
783: Stack Pointer

Provate, ad esempio:

Poke 781,1: Sys 58251

In seguito, come argomento della Poke, inserite altri valori (sempre compresi tra 1 e 30).

Ecco la tabella degli errori:

Codice Decim.	Codice Esadec.	Errore
1	$01	Too Many Files
2	$02	File Open
3	$03	File Not Open
4	$04	File Not Found
5	$05	Device Not Present
6	$06	Not Input File
7	$07	Not Output File
8	$08	Missing File Name
9	$09	Illegal Device Number
10	$0A	Next Without For
11	$0B	Syntax
12	$0C	Return Without Gosub
13	$0D	Out Of Data
14	$0E	Illegal Quantity
15	$0F	Overflow
16	$10	Out Of Memory
17	$11	Undef'd Statement
18	$12	Bad Subscript
19	$13	Redim'd Array
20	$14	Division By Zero
21	$15	Illegal Direct
22	$16	Type mismatch
23	$17	String Too Long
24	$18	File Data
25	$19	Formula Too Complex
26	$1A	Can't Continue
27	$1B	Undef'd Function
28	$1C	Verify
29	$1D	Load
30	$1E	Break

Sinora abbiamo visto in dettaglio come "prendere" un parametro numerico tra 0 e 255, controllarlo, usarlo, generare errori, e così via; ma, nella pratica, spesse volte il parametro può essere un numero maggiore di 255, oppure una stringa.

Vediamo, una per una, le routine già disponibili nelle Rom del C/64.

Lettura di un numero compreso tra 0 e 65535

La parte di S.O. interessata a svolgere questo compito è composta da due routine. La prima valuta l'espressione numerica ed è posta a partire da $AD8A, mentre la seconda ($B7F7) è quella che effettivamente legge e interpreta il numero.

Le due routine vanno chiamate nell'ordne descritto. Il valore letto (tra 0 e 65535) viene automaticamente scomposto in due byte (alto / basso) e quindi immesso nelle locazioni di memoria $14 $15.

Vediamo un esempio:

** Routine 4 **

```
JSR $AEFD ;Controllo virgola
JSR $AD8A ;Valuta il parametro
JSR $B7F7 ;Legge il numero
  ;compreso tra 0 e 65535
  ;e lo mette in $14/$15
LDA $14   ;Trasferisce il numero
STA 251   ;In 251/252
LDA $15
STA 252
RTS
```

Impartendo, da Basic, un comando del tipo "SYS ind,num" dove "ind" è l'indirizzo di partenza della routine e "num" è il numero da passare, la routine LM leggerà questo numero e lo trasferirà nelle locazioni 251 e 252, da cui può essere facilmente prelevato per qualsiasi scopo, anche da basic, tramite un Print Peek (251) + Peek (252)*256.

Ma perchè trasferiamo il numero dalle locazioni $14/$15 in 251 e 252? Non potremmo utilizzarlo direttamente dalle prime due?

Si potrebbe, ma è buona norma salvare in qualche zona di memoria "protetta" dal S.O. il numero letto, dato che le locazioni $14/$15 sono usate da innumerevoli routine dello stesso S.O. e si rischia di perderle ad una successiva chiamata.

Lettura di una stringa (fino a 255 caratteri)

Anche questa volta è necessario richiamare due routine del S.O: la prima posta in $AD9E, che valuta un'espressione qualsiasi, e la seconda a $B782, che legge un valore di tipo stringa.

Dopo la chiamata alle due routine, le locazioni $22/$23 conterranno il puntatore al primo carattere della stringa, mentre nel registro Y sarà contenuta la lunghezza della stringa stessa.

Vediamo, anche per questa routine, un esempio:

```
;* Routine 5 *

JSR $AEFD ;Virgola
JSR $AD9E ;Valuta l'espressione
JSR $B782 ;Legge una stringa,
 ;la fa puntare da $22/$23
 ;lungh.in Y
TYA       ;Trasferisce la lunghezza
 ;in Accumulatore
JMP $AB24 ;Stampa stringa
 ;da $22/$23 e lungh.in Acc
```

Al richiamo del programma (tramite SYS ind,"stringa"), verrà stampata su video la stringa inserita. E' buona norma, per la ragione spiegata prima, salvare il contenuto delle locazioni $22/$23 e quello di Y in qualche zona di memoria "protetta" dal S.O, specialmente se i dati serviranno molto tempo dopo il loro inserimento. Questa precauzione eviterà spiacevoli inconvenienti.

Controllo di un parametro numerico o stringa

Ci siamo occupati della lettura di parametri di cui conoscevamo la natura (numerici o stringa), ma come dobbiamo comportarci nel caso in cui non sapessimo di che tipo è il parametro inserito? Potremmo aver bisogno di passare un numero in certi casi, una stringa in altri, e svolgere diversi compiti a seconda del tipo di parametro inserito.

Naturalmente nel S.O. esiste una routine adatta a svolgere questo compito: si chiama "Evaluate Expression" ed è posta a partire dall'indirizzo $AD9E.

Questa esegue una valutazione del parametro letto e, a seconda del tipo, influenza una particolare locazione di memoria ($0D) che verrà messa a 0 se il parametro letto è di tipo numerico, mentre verrà posta a 255 se è di tipo stringa.

Una volta valutato il parametro bisogna controllare $0D per decidere in quale parte di programma saltare per gestire il dato stesso.

Dopo aver richiamato la evaluate expression ($AD9E), per leggere il parametro richiesto non sarà più necessario eseguire il primo JSR, dato che rieseguirebbe un compito già svolto (dando, per di più un errore), ma basterà soltanto eseguire il secondo, cioè solo JSR $B7F7 per i numeri e solo JSR $B782 per le stringhe.

Anche per questa routine è proposto un esempio:

```
;* Routine 6*

JSR $AEFD ;Virgola
JSR $AD9E ;Evaluate Expression
LDA $0D   ;Carica in A il flag
 ;numero/stringa
BEQ num   ;Se è 0, esegue parte
 ;di prog. per num.
JSR $B782 ;Se no continua
 ;con gest. stringa
.........
RTS
num:
 JSR $B7F7 ;Legge numero
 ......... ; Gest. Numero
 RTS
```

Una cosa da notare è che tutte le routine descritte sono protette da qualsiasi tipo di errore di inserimento, ossia verranno emessi appositi messaggi se si inserisce un numero al posto di una stringa, se il numero è troppo grande o negativo, se la stringa è troppo lunga, eccetera.

I parametri potranno, inoltre, essere dati in modo esplicito (ossia numeri e stringhe) oppure sotto forma di variabili o espressioni. Saranno lecite ad esempio le seguenti istruzioni:

```
SYS ind,1573
SYS ind,A
SYS ind,COS(B)+A*2/C

SYS ind,"prova"
SYS ind,A$
SYS ind,F$+LEFT$(H$,2)+"ciao"
```

Naturalmente, i comandi si possono usare, oltre che in forma diretta, anche all'interno di un programma.

Abbiamo visto come trattare routine che necessitano di un numero di parametri fisso, ma a volte il numero di parametri può risultare variabile.

In pratica, se si vuole modificare un solo parametro, non è necessario scrivere tutti gli altri, ma solo il parametro da modificare, risparmiando così tempo e byte di memoria.

Prendiamo come esempio la routine 1. Questa accetta in ingresso SOLO due parametri: volendo modificare il solo colore di sfondo siamo costretti ad inserire anche l'altro valore. Per eliminare l'inconveniente la strada da seguire è piuttosto semplice:

a) Leggere il primo parametro ed immetterlo nella locazione del colore sfondo
b) Il Prossimo carattere è una virgola?
c) No: Fine dei parametri e ritorno al basic
d) Si: Lettura di un altro parametro e cambio colore bordo

La cosa "nuova" da fare è il controllo della virgola. Perchè "nuova", se ne abbiamo già parlato? La routine descritta ($AEFD) NON si può utilizzare in questo caso poichè controlla la presenza di una virgola ma, in caso contrario, stampa un Syntax error, arrestando l'esecuzione del programma; abbiamo invece bisogno che il programma continui anche in caso di assenza della virgola.

Questa volta non esiste (o quasi) una routine già pronta e dovremo provvedere "manualmente" al controllo della virgola. La comprensione risulta più facile esaminando direttamente una routine:

```
;* Routine 7 *

JSR $AEFD ;Contr. prima virgola
JSR $B79E ;Lettura val. sfondo
STX 53281 ;Cambio colore
JSR $0079 ;Lettura prossimo carattere
CMP #$2C  ;E' una virgola?
BNE out   ;No: Torna al basic
JSR $AEFD ;Si: "salta" la virgola
JSR $B79E ;Leggi val. bordo
STX 53280 ;Cambia colore
out:
 RTS    ;Ritorno al basic
```

La parte di routine che interessa maggiormente è quella relativa alle due istruzioni JSR $0079 e CMP

MASTERBIT VENDITA PER CORRISPONDENZA

STAR NL 10

80 col. 120 cps bidirez. NLQ foglio singolo e cont. dedicate per COMMODORE PC/IBM APPLE II C Macintosh Sinclair QL L. 670.000 + interfaccia.

STAR D 10

80 col. 160 cps int. seriale e parallela L.480.000.

STAR D 15

132 col. 160 cps int. seriale e parallela L. 700.000.

GEMINI 160

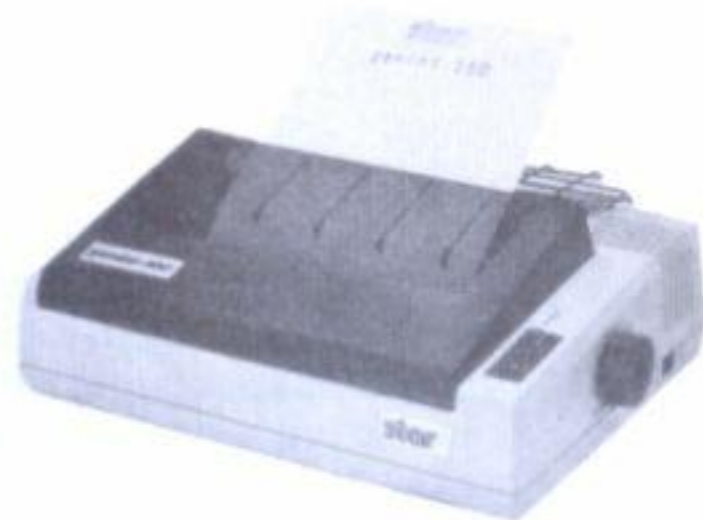
80 col. 160 cps bidirez. foglio sing. e cont. con interf. Centronics o IBM
.... L. 555.000
con interf. Centronics e seriale
.... L. 645.000
con interf. SECUS per C64/128
.... L. 685.000

AMSTRAD PC/IBM Comp. 1512

80/86 - 8 MHz - 512K - Drive 360K - Monitor - Interf. Parallela e Seriale - MS/DOS 3.2 GEM-DESKTOP - GEMPAINT - Basic 2.

Tutto a L. 1.399.000
con IVA e trasporto, 6 mesi di garanzia

Varie versioni: 2 Drive - 1 Drive + 1 Hard 10 o 20 M - monitor colori

AMSTRAD DMP 3000
80 col. 100 cps NLQ .... L. 549.000
AMSTRAD DMP 4000 .... L. 990.000
ESPANSIONE 640K .... L. 150.000
SCHEDA 20Mb .... L. 990.000

QL SINCLAIR 128K
L. 429.000

QL versione JS con 2 microdrive, alimentatore, manuale in inglese, manuale in italiano, per la gestione dei 4 programmi, cavetti, 4 cartucce con i quattro programmi gestionali. QUILL - ARCHIVE - EASEL - ABACUS, una cartuccia con 6 giochi originali più un super copiatore per MDV e FLP.

COMPUTERS

AMSTRAD IBM comp. conf. base .... L. 1.399.000
CONDOR PC/XT .... L. 1.950.000
640 doppio CLOCK 2 drive scheda grafica o colore - monitor monoc. sist. oper. MS DOS - tastiera italiana
PC EXPRESS .... L. 1.765.000
256K 1 drive da 360k scheda graf. col. scheda Hercules
PC WORD PROCESSOR AMSTRAD
256K 1 drive 3" monitor stampante NLQ .... L. 1.350.000
512K 2 drives 3" monitor stampante NLQ .... L. 1.595.000
QL SINCLAIR .... L. 429.000
SPECTRUM PLUS 48K .... L. 260.000
alimen. man. in ingl. ed in ital., cavetti, 5 progr. supercopiatore
SPECTRUM 128K 2 cassette con giochi .... L. 349.000
SPECTRUM PLUS 2 128K .... L. 429.000
registratore incorporato 1 joystick e 6 giochi
PC MICROTEK 256K .... L. 1.950.000
2 drives scheda graf. col. scheda stamp., monitor fosf. verdi
PHILIPS MSX 1 VG 8020 .... L. 425.000
PHILIPS MSX 2 NMS 8220 .... L. 670.000
PHILIPS MSX 2 con drive incorporato .... L. 1.150.000
COMMODORE 64 prima vers. con registratore .... L. 430.000
COMMODORE 64 seconda vers. new .... L. 450.000
COMMODORE 128 .... L. 600.000
AMIGA 512K .... L. 2.500.000
drive 3"1/2 monitor col. garanzia italiana
COMMODORE 128 D .... L. 1.250.000
128K drive 5"1/4 sist. oper. italiano
COMMODORE C128 .... L. 510.000
ATARI 520 ST 512K mouse .... L. 739.000
ATARI 520 STM 512K mouse modulatore TV .... L. 799.000

STAMPANTI

SMITH CORONA .... L. 320.000
80 col. 100 cps per Spectrum
SHINWA VP 8100 .... L. 440.000
80 col. 100 cps. semigrafica int. seriale o parallela
MANNESMANN TALLY MT80 + ... L. 549.000
80 col. 100 cps bidirez. interf. Centronics
MANNESMANN TALLY MT 80 PC .... L. 630.000
80 col. 120 cps bidirez. IBM/comp.
MANNESMANN TALLY MT 85 .... L. 849.000
80 col. 180 cps NLQ bidirez. interf. parallela o seriale IBM compatibili
MANNESMANN TALLY MT 86 .... L. 1.050.000
136 col. 180 cps NLQ bidirez. inter. parall. o ser. IBM comp.
MANNESMANN TALLY MT 290 .... L. 1.870.000
132 col. 200 cps NLQ interf. parallela
CENTRONICS GLP .... L. 549.000
80 col. 100 cps interf. parallela
CENTRONICS 220 .... L. 949.000
136 col. 180 cps NLQ interf. parallela e seriale
CBM MPS 803 - 80 col. 80 cps. .... L. 470.000
CBM MPS 1000 - 80 col. 100 cps. .... L. 649.000
CITIZEN 120 D .... L. 520.000
80 col. 100 cps per C 64 e C 128
SEIKO DPU 40 .... L. 290.000
40 col. 24 cps termica interf. parallela
PHILIPS W0020 80 col. .... L. 440.000
PHILIPS VW0030 80 col. 100 cps NLQ .... L. 655.000
EPSON P 40 port. term. 40 col. 40 cps int. seriale .... L. 340.000
EPSON LX 86 80 col. 120 cps NLQ per PC .... L. 720.000
EPSON LX 90 80 col. 100 cps NLQ per C 64/128 .... L. 750.000
COPAL 5500 .... L. 1.050.000
136 col. 180 cps 4k buffer NLQ

MONITORS

HANTAREX BOXER 12 .... L. 229.000
12" fosf. verdi alta risoluzione
HANTAREX 14 .... L. 499.000
14" colore standard risoluz. 80 col.
FENNER per C 64/128 .... L. 250.000
SLIP STREAM .... L. 465.000
14" colore standard risoluz. 40 col.

MODEM

PC CARD Modem V21/V23 .... L. 449.000
MODEM 300 .... L. 180.000
300 baud full duplex seriale
MULTISTANDARD .... L. 280.000
300 + 300 baud 300/1200 per VIDEOTEL AUTOMATICO
MAGNETOPLAST 300 baud .... L. 130.000
MODEM senza cuffia per COMMODORE .... L. 130.000
300/1200 baud con cavo, floppy, manuale
MODEM PHONE 1100 .... L. 345.000
con telefono 300/1200 baud full e half duplex per PC cavo ser.
MODEM 130 .... L. 245.000
300 baud full duplex per PC cavo seriale
MODEM 230 .... L. 345.000
300 baud funz. autom. per PC Comp. HAYES cavo seriale
MODEM PHONE 303 .... L. 230.000
il più economico con protocollo CCITT V 21 - 300 baud cavo ser.
MODEM 1200 RF .... L. 560.000
CCITT V21/V22 BELL 103/202 - 300/600/1200 baud può allacciarsi a qualunque sistema di ricetrasmittente, radiotelefoni - OM - CB.

JOYSTICK

DATALINE standard 9 PIN D .... L. 14.000
DATALINE MINI .... L. 48.000
SPECTRAVIDEO QS II .... L. 16.000
SPECTRAVIDEO QS IV .... L. 20.000
SPECTRAVIDEO QS IX .... L. 25.000
MAGNUM per C 16 .... L. 23.000

SINCLAIR QL

QL 640K .... L. 689.000
DRIVE da 3"1/2 + interf. per QL professionale .... L. 630.000
oltre 700K formattati alim. incorp.
DOPPIO DRIVE come sopra in unico contenitore .... L. 899.000
ESPANSIONE QL 640K .... L. 260.000
2 ROM JS (trasf. il QL da JM a JS) .... L. 80.000
CONVERTITORE RS 232 Centronics con cavo per QL .... L. 99.000
CAVO di collegamento QL/RS232 .... L. 39.000
CAVO JOYSTICK per QL .... L. 19.000
CAVO SER 1 per QL .... L. 15.000
TOOLKIT II su ROM .... L. 60.000
TUTTI I PEZZI DI RICAMBIO: es. Contattiera .... L. 26.000
SUPER MOUSE QL .... L. 185.000
BOX per Microdrive .... L. 15.000
Copritastiera per QL .... L. 12.000

SINCLAIR SPECTRUM

Interfaccia 1 .... L. 155.000
Microdrive .... L. 120.000
Interfaccia 1 + Microdrive + Cartuccia dimostr. .... L. 260.000
Interfaccia Beta .... L. 320.000
Trasformazione da Spectrum a Spectrum Plus .... L. 105.000
Convertitore RS232 Centronics .... L. 99.000
Interfaccia Centronics su ROM .... L. 99.000
Interfaccia RS 232 .... L. 99.000
Interfaccia joystick tipo Kempston 1 presa .... L. 25.000
Interfaccia joystick tipo Kempston 2 prese .... L. 35.000
Interfaccia parlante CURRAH - Parla italiano .... L. 65.000
TRISLOT presa tripla L. 27.000
Discipline interface .... L. 195.000
Interfaccia 1 .... L. 140.000
Multiface 1 .... L. 99.000
Multiface 1 128k .... L. 105.000
Confezione da 4 cartucce per Microdrive .... L. 24.000
TUTTI I PEZZI DI RICAMBIO: e s. Ula .... L. 38.000

VARIE

Tutti gli articoli TOSHIBA
Nastri inchiostrati PELIKAN per stampanti
Tutti gli articoli EPSON
DISPONIBILI 1200 programmi per PC/comp.
Floppy 3"1/2 - VERBATIM DF/DD .... L. 7.000
Duplicatori per cassette per C 64 .... L. 24.000
MOUSE per C 64/C 128 .... L. 110.000
VIDEO CASSETTE SUPER HIGH GRADE da 120 .... L. 8.000
VIDEO CASSETTE SUPER HIGH GRADE da 180 .... L. 9.500
Registratore PHILIPS D6450 .... L. 110.000
Drive PHILIPS VY0010 .... L. 610.000
Dischetti NASHUA DF/DD 5"1/4 .... L. 2.600
HARD DISK - accessori e periferiche per compatibili IBM

AVVERTENZE - Tutti i prezzi sono comprensivi di IVA e spese postali, per ordini inferiori alle 50.000 lire aggiungere L. 8.000 per contributo spese di spedizione - pagamento contrassegno al ricevimento del pacco. (È gradito il contatto telefonico). **SCONTI QUANTITÀ.**

ORDINI TELEFONICI
ORE 8.30/20.30 - Tel. 06/5611251

Garanzia 48H - La MASTERBIT si impegna a sostituire quegli articoli riscontrati malfunzionanti entro 48 ore dal ricevimento, ogni articolo è fornito di regolare garanzia.

MASTERBIT Viale dei Romagnoli 35 - 00121 OSTIA LIDO RM - CAS. POST. 3016

#$2C. Il JSR richiama una routine presente nella pagina zero del C/64 che provvede a leggere il carattere indirizzato dal puntatore (SENZA aggiornarlo) ed a porlo nell'accumulatore (A). In seguito lo stesso carattere viene confrontato con il numero $2C, codice della virgola. Nel caso non sia una virgola, verrà eseguito il salto (BNE out) al termine del programma senza stampare errori anche in caso di assenza del secondo parametro. In caso contrario si DOVRA' richiamare ugualmente la routine di controllo della virgola ($AEFD), dato che questa deve aggiornare il puntatore del carattere, poi si potrà continuare con la lettura e la normale gestione del successivo parametro.

La sintassi con questo tipo di routine è la seguente:

SYS ind.CS[,CB]

Le parentesi quadre indicano che il parametro CB è opzionale. La sintassi può quindi avere due forme:

SYS ind,CS
SYS ind,CS,CB

Il listato proposto in queste pagine, che carica in memoria la routine in LM, serve per cambiare i colori di sfondo, di bordo e del carattere, ed accetta, quindi, tre parametri. Naturalmente viene effettuato un controllo e non risulta necessario inserirli tutti e tre.

SCHEDA TECNICA

Software LM applicativo di tipo didattico.

Idoneo per computer C/64 e non facilmente adattabile ad altri computer Commodore

Richiede una certa conoscenza del linguaggio macchina e viene consigliato ai lettori che vogliano approfondire gli argomenti tipici dell'Assembly.

Anche il programma pubblicato in queste pagine è contenuto nel disco "Directory" di questo mese.

```
100 REM ROUTINE LM PER CAMBIARE
110 REM COLORE ALLO SFONDO, AL
120 REM BORDO ED AI CARATTERI
130 REM BY GIANCARLO MARIANI
140 :
150 REM SINTASSI:
160 :
170 REM SYS 826,CS[,CB[,CC]]
180 :
190 REM CS=COLORE SFONDO
200 REM CB=COLORE BORDO
210 REM CC=COLORE CARATTERI
220 :
230 REM LE PARENTESI QUADRE IND
    ICANO
240 REM CHE IL PARAMETRO E' OPZ
    IONALE
250 REM CIOE' PUO' ESSERE OMESS
    O
260 :
270 :
280 CK=0:FOR T=0 TO 48:READ A
290 POKE 826+T,A:CK=CK+A:NEXT
300 IF CK=4226 THEN PRINT "TUTT
    O OK!":GOTO460
310 PRINT CHR$(147);"ERRORE NEI
     DATI !":END
320 :
330 REM DATI PER ROUTINE LM
340 :
350 DATA 032,091,003,142,033
360 DATA 208,032,121,000,201
370 DATA 044,208,019,032,091
380 DATA 003,142,032,208,032
390 DATA 121,000,201,044,208
400 DATA 006,032,091,003,142
410 DATA 134,002,096,032,253
420 DATA 174,032,158,183,224
430 DATA 016,016,001,096,162
440 DATA 014,108,000,003
450 :
460 PRINT"SYS 826,CS,CB,CC"
470 END
```

```
** DISASSEMBLATO LM **

ROUTINE PRINCIPALE (MAIN)

$033A JSR $035B  ; PRENDE IL PRIMO PAR.
      STX $D021  ; LO METTE NEL COLORE SFONDO
      JSR $0079  ; CONTROLLA CAR. CORRENTE
      CMP #$2C   ; E' UNA VIRGOLA
      BNE $035A  ; NO:FINISCI
      JSR $035B  ; SI:PRENDI ALTRO PARAM.
      STX $D020  ; METTILO IN COL. BORDO
      JSR $0079  ; CONTROLLA CAR. CORRENTE
      CMP #$2C   ; E' UNA VIRGOLA
      BNE $035A  ; NO:FINISCI
      JSR $035B  ; SI:PRENDI TERZO PARAM.
      STX $0286  ; METTILO IN COL. CARATT.
$035A RTS        ; TORNA AL BASIC

ROUTINE CHE PRENDE UN PARAM.
E CONTROLLA SE E' >15

$035B JSR $AEFD  ; CONTR. E SALTA VIRGOLA
      JSR $B79E  ; PONE IN X NUM. TRA 0 E 255
      CPX #$10   ; X E' >=16
      BPL $0366  ; SI:ERRORE
      RTS        ; NO:TORNA AL MAIN

ROUTINE DI ERRORE

$0366 LDX #$0E   ; PUNTA A ILLEGAL QUANTITY
      JMP ($0300) ; STAMPA L'ERRORE
```

Minicopiatore di programmi

Un programma di utilità, in LM, per chi possiede un drive e vuole cimentarsi in Assembly

di Paolo Agostini

Uno dei migliori sistemi a disposizione per l'apprendimento è quello dell'imitazione. Gli abili maestri artigiani del Medioevo affermavano che colui che voleva apprendere l'arte doveva "rubarla con l'occhio" e ciò è vero anche oggi, nell'epoca del computer. E' proprio per questa ragione che stavolta vi presentiamo un programma che, benchè breve, ha due scopi dichiarati.

Il primo scopo è quello di soddisfare i bisogni immediati di coloro che, spesso, si trovano nella necessità di trasferire programmi in linguaggio macchina da un disco ad un altro. Com'è noto i comandi del Commodore 64 consentono il trasferimento mediante il Load e il successivo Save soltanto di programmi in Basic o che iniziano nella zona di Ram riservata al Basic (vale a dire dalla locazione di memoria 2049, in esadecimale $0801). Il minicopiatore di queste pagine permetterà, invece, di trasferire qualsiasi programma da un disco ad un altro, qualunque sia la locazione di inizio. E' bene precisare, però, che è possibile copiare soltanto file che nella directory vengono indicati come PRG.

Il secondo fine è invece quello di fornire, a coloro che stanno muovendo i primi passi nel linguaggio macchina, grazie al disassemblato commentato del programma, una pratica applicazione di numerose, piccole routine, utili per altri programmi. Tra le altre facciamo notare la routine di Input che serve al copiatore per conoscere il nome del programma da copiare; vi è poi l'utilizzo delle routine del Kernal per aprire e chiudere file e per leggerne il contenuto; c'è anche una routine che consente di leggere lo status del disk drive e di stamparlo sullo schermo.

Troverete inoltre la routine che aspetta l'input di un carattere dalla tastiera e che corrisponde alla riga di Basic:

100 get a$:if a$="" then 100

mediante l'utilizzo della routine GETIN del Kernal. C'è la dimostrazione di come utilizzare la routine che inizia alla locazione $AB1E per la stampa di stringhe: la stringa posta in memoria non deve essere più lunga di 255 caratteri e l'ultimo carattere deve obbligatoriamente essere uno zero. Dopo aver caricato l'Accumulatore con il LSB e il registro Y col MSB della locazione in cui la stringa è collocata, basta saltare alla routine $AB1E per stampare la stringa di caratteri sullo schermo.

Vi sono, insomma, parecchie cosette da studiare, imitare e magari perfezionare: un assioma fondamentale dell'informatica è che non esiste un programma che non possa essere ulteriormente perfezionato!

I tedeschi dicono: "Uebung macht den Meister", l'esercizio fa il maestro. Esercitatevi, quindi, e non lasciatevi abbattere dalle difficoltà iniziali.

Come usare il programma

Il caricatore in Basic serve, come al solito, a creare sul vostro disco il programma definitivo in linguaggio macchina; controlla i dati, riga per riga, e se vi sono discordanze provvede a segnalare le righe alle quali avete probabilmente commesso un errore di copiatura. Se il programma si blocca segnalando un errore nel loop che legge i DATA, ciò significa che avete inserito un numero maggiore di 256.

Per sapere quale sia la riga alla quale avete commesso l'errore basterà dare il seguente comando diretto:

*print peek (64)*256 + peek (63)*

e dovrebbe apparire sullo schermo la riga di DATA incriminata.

Una volta che abbiate il vostro minicopiatore su disco, sarà sufficiente caricarlo e dargli il RUN, come se si trattasse di un qualsivoglia programma in Basic.

```
100 REM  P.AGOSTINI: MINICOPIAT
    ORE DI PROGRAMMI
110 DATA  011,008,194,007,158,0
    50,048,054,049,000,067
120 DATA  000,000,032,162,008,1
    69,253,160,008,032,056
130 DATA  030,171,160,000,132,0
    02,032,207,255,153,118
140 DATA  121,009,200,201,013,2
    08,245,136,192,000,045
150 DATA  240,240,152,234,162,1
    21,160,009,032,189,003
160 DATA  255,169,002,162,008,1
    60,000,032,186,255,205
170 DATA  032,192,255,032,193,0
    08,165,002,208,091,154
180 DATA  162,002,032,198,255,0
    32,228,255,160,000,044
190 DATA  145,251,165,144,208,0
    06,032,179,008,076,190
200 DATA  076,008,165,251,133,2
    53,165,252,133,254,154
210 DATA  032,162,008,169,078,1
    60,009,032,030,171,083
220 DATA  032,228,255,201,013,2
    08,249,169,002,162,239
230 DATA  008,160,001,032,186,2
    55,032,192,255,162,003
240 DATA  002,032,201,255,032,1
    86,008,032,179,008,167
250 DATA  165,251,197,253,208,2
    44,165,252,197,254,138
260 DATA  208,238,032,186,008,0
    32,162,008,076,193,119
270 DATA  008,169,138,133,251,1
    69,009,133,252,169,151
280 DATA  002,032,195,255,032,2
    04,255,096,230,251,016
290 DATA  208,002,230,252,096,1
    60,000,177,251,076,172
300 DATA  210,255,169,000,133,1
    44,169,013,032,210,055
310 DATA  255,169,008,133,186,0
    32,180,255,169,111,218
320 DATA  133,185,032,150,255,1
    64,144,208,026,032,049
330 DATA  165,255,201,048,240,0
    12,133,002,076,239,091
340 DATA  008,164,144,208,010,0
    32,165,255,032,210,204
350 DATA  255,201,013,208,242,0
    32,186,255,032,204,092
360 DATA  255,096,147,013,077,0
    73,078,073,045,067,156
370 DATA  079,080,073,065,084,0
    79,082,069,032,068,199
380 DATA  073,032,080,082,079,0
    71,082,065,077,077,206
390 DATA  073,032,068,065,032,0
    68,073,083,067,079,128
400 DATA  013,066,089,032,080,0
    65,079,076,079,032,099
410 DATA  065,071,079,083,084,0
    73,078,073,032,080,206
420 DATA  068,032,056,054,013,0
    13,078,079,077,069,027
430 DATA  032,080,082,079,071,0
    82,065,077,077,065,198
440 DATA  058,032,000,013,073,0
    78,084,082,079,068,055
450 DATA  085,082,082,069,032,0
    78,085,079,086,079,245
460 DATA  032,068,073,083,067,0
    79,032,069,032,080,103
470 DATA  082,069,077,069,082,0
    69,032,082,069,084,203
480 DATA  085,082,078,013,000,0
    00,000,000,000,000,002
490 DATA  000,000,000,000,000,0
    00,000,000,000,000,000
500 DATA  000,000,000,000,095,0
    95,047,047,015,001,044
510 FOR I= 2049  TO  2444  STEP
    10:Z=0
520 FOR J=0 TO 9:READ X:CK=CK+X
    :Z=(Z+X) AND 255:NEXTJ
```

```
530 READ W:RIGA=PEEK(63)+PEEK(6
    4)*256
540 IF Z<>W THEN PRINT"ERRORE N
    ELLE RIGHE";RI-10;"-";RI+10
550 NEXTI:IF CK<>41568 THEN PRI
    NT"MANCA QUALCHE RIGA DI DA
    TI":END
560 PRINT"DATI ESATTI: INSERIRE
     UN DISCO E":PRINT"POI PREM
    ERE UN TASTO"
570 POKE 198,0:WAIT 198,1:POKE
    198,0:PRINT"OK":RESTORE
580 OPEN 15,8,15,"I0":OPEN 8,8,
    8,"MINICOPIER,P,W"
590 INPUT#15,E1,E1$:IF E1 THEN
    PRINT"ERRORE DISCO";E1;E1$:
    CLOSE 8:CLOSE 15:END
600 PRINT#8,CHR$(1);CHR$(8);:RE
    M  ATTENZIONE AL PUNTO E VI
    RGOLA!!!
610 FOR I= 2049  TO  2444  STEP
     10
620 FOR J=0 TO 9:READ X:PRINT#8
    ,CHR$(X);:REM  ATTENZIONE A
    L PUNTO E VIRGOLA!
630 NEXTJ:READ W:NEXTI
640 CLOSE 8:CLOSE 15:PRINT"SUL
    DISCO C'E' ORA IL TUO PROGR
    AMMA!"
```

```
  1 *
  2 * MINICOPIATORE DI PROGRAMMI
  3 * IN BASIC O LINGUAGGIO MACCHINA
  4 * DA DISCO A DISCO
  5 *
  6
  7 * indirizzi usati nell'assemblato
  8
  9 DEVNUM   -      $BA
 10 IECTALK  -      $FFB4
 11 SECADR   -      $B9
 12 SENDSEC  -      $FF96
 13 IECINP   -      $FFA5
 14 UNTALK   -      $FFBA
 15 SETLFS   -      $FFBA
 16 SETNAM   -      $FFBD
 17 OPEN     -      $FFC0
 18 CLOSE    -      $FFC3
 19 CHKIN    -      $FFC6
 20 CHKOUT   -      $FFC9
 21 CLRCHN   -      $FFCC
 22 CHROUT   -      $FFD2
 23 GETIN    -      $FFE4
 24 CHRIN    -      $FFCF
 25 AUX1     -      $FB
 26 AUX2     -      $FD
 27 STATUS   -      $90
 28 CR       -      13
 29 ERRFLAG  -      $02
 30
 31
 32          TR     ON          ;tronca righe di data
 33
 34 *-----------------------
 35
 36
 37          ORG    $0801       ;start dal BASIC
 38
 39
 40          DA     TWOBRK      ;link alla fine della riga
 41          DA     1986        ;numero di riga
 42          DFB    $9E         ;token di SYS
 43          TXT    '2061' ;locazione inizio prg in l.m.
 44          BRK                ;fine della riga BASIC
 45 TWOBRK   DA     0           ;indica fine programma in BASIC
 46          ERR    *-2061      ;se non corrisponde a 2061 errore!
 47
 48
 49 *-----------------------
 50
 51
 52          JSR    CLSFIL      ;chiude il file se aperto
 53
 54          LDA    #<MSG1      ;carica LSB del messaggio in A
 55          LDY    #>MSG1      ;carica MSB del messaggio in Y
 56          JSR    $AB1E       ;va alla routine di stampa righe
 57
 58          LDY    #0          ;azzera il contatore Y
 59          STY    ERRFLAG     ;azzera il FLAG di errore
 60 RD       JSR    CHRIN       ;va alla routine di INPUT
 61          STA    FILNAM,Y    ;e memorizza il nome del file
 62          INY                ;incrementa il contatore
 63          CMP    #CR         ;se = RETURN fine dell'input
 64          BNE    RD          ;altrimenti prendi carattere successivo
 65
 66
 67          DEY                ;decrementa di 1 il contatore
 68 *                           ;(per evitare di usare RETURN
 69 *                           ;come parte del nome del file!)
 70          CPY    #0          ;confronta lunghezza del file
 71          BEQ    RD          ;se =0 aspetta input
 72          TYA                ;trasferisce lunghezza del
 73          NOP                ;file nell'accumulatore
 74          LDX    #<FILNAM    ;LSB del nome in X
 75          LDY    #>FILNAM    ;MSB del nome in Y
 76          JSR    SETNAM      ;set up file name !
 77          LDA    #$02        ;apri il canale 2
 78          LDX    #$08        ;con la periferica 8
 79          LDY    #$00        ;indirizzo secondario 0
 80          JSR    SETLFS      ;set up logical file number!
 81          JSR    OPEN        ;routine di apertura canale
 82
 83          JSR    READST      ;legge il canale di errore
 84          LDA    ERRFLAG     ;se vi e' un errore esce dal prg
 85          BNE    CLSFIL
 86
 87          LDX    #$02        ;corrisponde al comando BASIC
 88          JSR    CHKIN       ;CMD2
 89 LOOP     JSR    GETIN       ;prende un carattere
 90          LDY    #$00        ;usa Y come puntatore per
 91          STA    (AUX1),Y    ;memorizzare il carattere
 92          LDA    STATUS      ;se ST non segnala fine del file
 93          BNE    INPUT       ;continua a caricare
 94          JSR    INCREASE    ;incrementa puntatore $FB/$FC
 95          JMP    LOOP        ;e continua
 96
 97 *-----------------------
 98
 99 INPUT    LDA    AUX1        ;se tutto e' terminato
100          STA    AUX2        ;mette in $FD/$FE il valore
101          LDA    AUX1+1      ;della locazione finale
102          STA    AUX2+1      ;del file appena caricato
103          JSR    CLSFIL      ;e chiude il canale col drive
104          LDA    #<MSG2      ;stampa il messaggio
105          LDY    #>MSG2      ;che chiede di inserire
106          JSR    $AB1E       ;il nuovo disco
107
108 INPUT1   JSR    GETIN       ;aspetta la pressione
109          CMP    #CR         ;del tasto RETURN
110          BNE    INPUT1      ;
111          LDA    #$02        ;apre un canale
112          LDX    #$08        ;col drive
113          LDY    #$01        ;(il nome del file resta
114          JSR    SETLFS      ;invariato!!)
115          JSR    OPEN        ;apre
116          LDX    #$02        ;corrisponde
117          JSR    CHKOUT      ;al comando NASIC CMD2!
118 INPUT2   JSR    OUTPUT      ;routine che invia 1 carattere
119          JSR    INCREASE    ;incrementa i puntatori
120          LDA    AUX1        ;confronta se valore max
121          CMP    AUX2        ;e' stato raggiunto
122          BNE    INPUT2      ;altrimenti dirama
123          LDA    AUX1+1      ;
124          CMP    AUX2+1      ;
125          BNE    INPUT2      ;
126          JSR    OUTPUT      ;routine che invia 1 car.
127          JSR    CLSFIL      ;chiude il file
128          JMP    READST      ;legge lo stato del drive
129
130 *-------------------------
131
132 CLSFIL   LDA    #<BUFFER    ;mette il valore delle
133          STA    AUX1        ;locaz.iniz.del buffer
134          LDA    #>BUFFER    ;nei puntatori $FB/$FC
135          STA    AUX1+1      ;
136          LDA    #$02
137          JSR    CLOSE       ;chiude il canale 2
138          JSR    CLRCHN      ;clear channel !
139          RTS
140
141 *-------------------------
142
143 INCREASE INC    AUX1        ;$FB/$FC = $FB/$FC + 1
144          BNE    INCR1
145          INC    AUX1+1
146 INCR1    RTS
147
148 *-------------------------
149
150 OUTPUT   LDY    #$00        ;invia 1 carattere
151          LDA    (AUX1),Y    ;al disk drive dopo averlo
152          JMP    CHROUT      ;prelevato dalla memoria
153
154 *-------------------------
155
156 READST   LDA    #0          ;routine di lettura status drive
157          STA    STATUS      ;azzera la variabile ST
158          LDA    #CR         ;stampa un RETURN
159          JSR    CHROUT      ;sullo schermo
160          LDA    #8          ;periferica numero otto
161          STA    DEVNUM      ;nell'opportuna locazione
162          JSR    IECTALK     ;routine di TALK del BUS
163          LDA    #$6F        ;indirizzo secondario 15 + 6*16
164          STA    SECADR
165          JSR    SENDSEC     ;routine che invia indir.sec.
166          LDY    STATUS      ;controlla variabile ST
167          BNE    LOOPEND     ;IF ST <> 0 THEN END
168          JSR    IECINP      ;prende un carattere dal BUS
169          CMP    #'0         ;lo confronta con lo zero
170          BEQ    STLOOP1     ;se =0 tutto OK!
171          STA    ERRFLAG     ;altrimenti errore !
172          JMP    STLOOP1     ;salta al loop di lettura status
173
174 STLOOP   LDY    STATUS      ;IF ST <> 0 THEN END
175          BNE    LOOPEND     ;
176          JSR    IECINP      ;prende un carattere dal BUS
177 STLOOP1  JSR    CHROUT      ;e lo stampa sullo schermo
178          CMP    #CR         ;lo confronta con RETURN
179          BNE    STLOOP      ;se <> return continua
180 LOOPEND  JSR    UNTALK      ;altrimenti termina
181          JSR    CLRCHN      ;pulisce il canale di comunicazione
182          RTS
183
184 *-------------------------
185
186 MSG1     DFB    147,13
187          TXT    'mini-copiatore di programmi da disco'
188          DFB    13
189          TXT    'by paolo agostini pd 86'
190          DFB    13,13
191          TXT    'nome programma:
192          DFB    0
193
194 MSG2     DFB    13
195          TXT    'introdurre nuovo disco e premere return'
196          DFB    13,0
197
198 *-------------------------
199
200          BRK
201 FILNAM   DFB    0,0,0,0,0,0,0,0
202          DFB    0,0,0,0,0,0,0,0,0
203
204 BUFFER   DFB    0
```

Quando uno schermo solo non basta

di Mario Breccia

Come tenere in memoria due schermate grafiche in alta risoluzione con il C/128

Il programma proposto è formato da quattro routine uguali (cambiano soltanto gli indirizzi) che consentono di avere in memoria due schermate grafiche Hi-Res oltre a quella normalmente disponibile nel C/128.

Il programma caricatore le alloca dalla locazione $1300 in poi, senza disturbare, quindi, un programma Basic eventualmente presente in memoria. Le routine utilizzano l'intera area di memoria posizionata da $9000 a $FC00. Non si è ritenuto necessario, infatti, abbassare i puntatori di fine programma.

Per trasportare il disegno presente nella pagina grafica standard in uno dei due buffer, basta impartire SYS

4864 per il primo, e SYS 5120 per il secondo.

Lo scambio inverso di dati tra uno dei buffer e l'area grafica è effettuabile (avendo allocato le routine a partire da $1300) con SYS 4935 per il primo buffer, e con SYS 5191 per il secondo.

Il trasferimento dati è velocissimo poiché sfrutta tutta la velocità del microprocessore 8510.

Oltre a settare il bit 1 della locazione $D030, infatti, attiva anche il blanking, accelerando ulteriormente le operazioni. Le routine sfruttano le locazioni di pagina zero allocate da $FC a $FF.

Agli esperti ricordiamo che ad ogni "uscita" dalle routine ci si troverà nel banco 15. L'entrata, invece, può essere effettuata da qualsiasi banco.

Come utilizzare le routine

Digitate dapprima il programma contenente le istruzioni Read...Data, salvatelo, verificatelo e attivatelo. Se lo avete trascritto correttamente comparirà il messaggio: "Tutto O.K."; in caso contrario effettuate i controlli del caso.

Digitate, quindi, il secondo listato (demo) e lanciatelo con il solito Run.

Il suo funzionamento è banale: dopo aver disegnato un quadrato, questo viene memorizato nel primo buffer; in seguito lo schermo viene cancellato per consentire il disegno di una circonferenza. La seconda parte del programma, mediante le Sys 4935 e 5191, permette di richiamare, rispettivamente, una delle due schermate memorizzate in precedenza.

Inutile dire che nella zona di memoria Ram, utilizzata dal programma LM, non devono esser presenti altre routine, pena l'inchiodamento del sistema.

SCHEDA TECNICA

Software LM applicativo di utilità.

Idoneo per computer C/128 in modo 40 colonne; non adattabile ad altri computer Commodore.

Consigliato agli appassionati di grafica che vogliano sfruttare le doti del C/128.

Anche i programmi pubblicati in queste pagine sono contenuti nel disco "Directory" di questo mese.

```
100 REM DEMO PER GESTIONE DEL BUFFER GRAFICO IN ALTA RISOLUZIONE PER C/128
110 REM BY MARIO BRECCIA - ROMA
120 :
130 REM E' INDISPENSABILE CARICARE DAPPRIMA LE ROUTINE LM!!!
140 :
150 PRINTCHR$(147)"ORA DISEGNO UN QUADRATO (PREMI UN TASTO)":GOSUB260:GOSUB270
160 BOX 1,20,60,100,140,0:REM DISEGNA UN QUADRATO
170 GOSUB260:SYS4864:GOSUB290:REM MEMORIZZA NEL PRIMO BUFFER
180 PRINTCHR$(147)"ORA DISEGNO UN CERCHIO (PREMI UN TASTO)":GOSUB260:GOSUB270
190 CIRCLE 1,160,100,40,40:REM DISEGNA CERCHIO
200 GOSUB260:SYS5120:REM MEMORIZZA NEL SECONDO BUFFER
210 GOSUB290:PRINTCHR$(147)"1- SCHERMO 1"
220 PRINT"2- SCHERMO 2":PRINT"3- FINE 3":GOSUB260:IFA$="3"THENEND
230 IFA$="1"THENSYS4935:GOSUB280:GETKEY B$:GOSUB290:REM RICHIAMA SCHERMO 1
240 IFA$="2"THENSYS5191:GOSUB280:GETKEYB$:GOSUB290:REM RICHIAMA SCHERMO 2
250 GOTO210
260 GETKEY A$:RETURN:REM ATTENDE PRESSIONE TASTO
270 GRAPHIC 1,1:RETURN:REM ENTRA IN MODO GRAFICO E CANCELLA
280 GRAPHIC 1,0:RETURN:REM ENTRA IN MODO GRAFICO SENZA CANCELLARE
290 GRAPHIC 0,1:RETURN:REM TORNA AL MODO TESTO E CANCELLA
300 END

READY.
```

```
100 REM SCREEN-BUFFER HI-RES PER C-128
110 REM BY MARIO BECCIA - ROMA
120 :
130 REM CHECKSUM = 10764
140 REM AREA DI MEMORIA UTILIZZATA: $1300 - $138D
150 REM $1300 - $138D PER LA PRIMA COPPIA DI ROUTINE ($9000 - $BC00)
160 REM $1400 - $148D PER LA SECONDA COPPIA DI ROUTINE ($C000 - $FC00)
170 :
180 A=0:X=0:CK=0:VI=1:FAST:POKE53280,0
190 DO:READ A
```

```
200 IF A=-1 THEN EXIT
210 POKE 4864+X,A:POKE4935+X,A:POKE5120+X,A:POKE 5191+X,A
220 VI=VI+1:IF VI=16 THEN VI=1
230 CK=CK+A:X=X+1:COLOR4,VI:LOOP
240 SLOW:READ CN:IF CN=CK THEN 350
250 PRINT"ERRORE DI TRASCRIZIONE":END
260 DATA 120,169,0,141,0,255,169,253,141
270 DATA 48,208,169,0,141,17,208,169,63
280 DATA 141,0,255,169,0,133,254,133,252
290 DATA 169,28,133,255,169,144,133,253
300 DATA 160,0,177,254,145,252,200,208
310 DATA 249,230,255,230,253,165,255,201
320 DATA 64,208,237,169,0,141,0,255,169
330 DATA 252,141,48,208,169,27,141,17
340 DATA 208,88,96,-1,10764
350 COLOR0,1:COLOR4,1:COLOR5,6
360 PRINT"TUTTO O.K.":POKE4973,252:POKE4975,254:POKE5152,192
370 POKE5223,192:POKE5229,252:POKE5231,254:END

READY.
```

```
Disassemblato commentato di Screen-Buffer per C/128
-----------------------------------------------------------

. 01300  78        sei          ;disabilita le interruzioni
. 01301  a9 00     lda #$00     ;manda nel banco 15
. 01303  8d 00 ff  sta $ff00    ;
. 01306  a9 fd     lda #$fd     ;attiva la modalita' FAST
. 01308  8d 30 d0  sta $d030    ;
. 0130b  a9 00     lda #$00     ;attiva il blanking
. 0130d  8d 11 d0  sta $d011    ;
. 01310  a9 3f     lda #$3f     ;manda nel banco 0
. 01312  8d 00 ff  sta $ff00    ;
. 01315  a9 00     lda #$00     ;
. 01317  85 fd     sta $fd      ;
. 01319  85 fc     sta $fc      ;
. 0131b  a9 1c     lda #$1c     ;
. 0131d  85 ff     sta $ff      ;setta i puntatori di pag. zero
. 0131f  a9 90     lda #$90     ;
. 01321  85 fd     sta $fd      ;
. 01323  a0 00     ldy #$00     ;azzera y
. 01325  b1 fe     lda ($fe),y  ;
. 01327  91 fc     sta ($fc),y  ;ciclo principale
. 01329  c8        iny          ;
. 0132a  d0 f9     bne $1325    ;se y<>0, vai a $1325
. 0132c  e6 ff     inc $ff      ;$ff=$ff+1
. 0132e  e6 fd     inc $fd      ;$fd=$fd+1
. 01330  a5 ff     lda $ff      ;carica l'acc. da $ff
. 01332  c9 40     cmp #$40     ;
. 01334  d0 ed     bne $1323    ;se acc. <>40 ,vai a $1323
. 01336  a9 00     lda #$00     ;manda al banco 15
. 01338  8d 00 ff  sta $ff00    ;
. 0133b  a9 fc     lda #$fc     ;riporta alla modalita' SLOW
. 0133d  8d 30 d0  sta $d030    ;
. 01340  a9 1b     lda #$1b     ;elimina il blanking
. 01342  8d 11 d0  sta $d011    ;
. 01345  58        cli          ;riabilita le interruzioni
. 01346  60        rts          ;ritorna al basic
-----------------------------------------------------------
```

Viaggio all'interno della traccia 18

Una chiacchierata per proseguire un discorso già iniziato in precedenza

di Paolo Agostini

E allora? Avete copiato il monitor per dischetto presentato sul N.36? Allora è giunto il momento di andare a vedere com'è fatta "dentro" la traccia 18.

Come si sono accorti tutti coloro che hanno abbandonato la lentissima Datassette per passare al disk drive, la cosa che distingue il disco rispetto al nastro è la preziosa presenza della DIRECTORY, ovverosia dell'indice del suo contenuto.

Questo "indice" è posizionato proprio nella traccia 18. Perchè proprio lì?

Semplice: perchè si trova proprio al centro del disco e ciò rende più veloce l'accesso alle tracce più esterne (cioè la traccia 1 e la traccia 35).

Nel lavorare con i file su disco (siano essi file programma, file sequenziali o di altro tipo) la testina deve "accedere" spesso alla Directory e bisogna fare in modo che, da qualunque posizione si trovi, raggiunga la traccia 18 nel minor tempo possibile.

Il posizionamento della testina di lettura, dalla traccia 18 alla traccia 1 come pure dalla traccia 18 alla traccia 35, avviene in circa 5 decimi di secondo, e il posizionamento dalla traccia 1 alla traccia 35 avviene in un tempo quasi doppio e tale periodo sarebbe inaccettabile per un disk drive degno di questo nome!

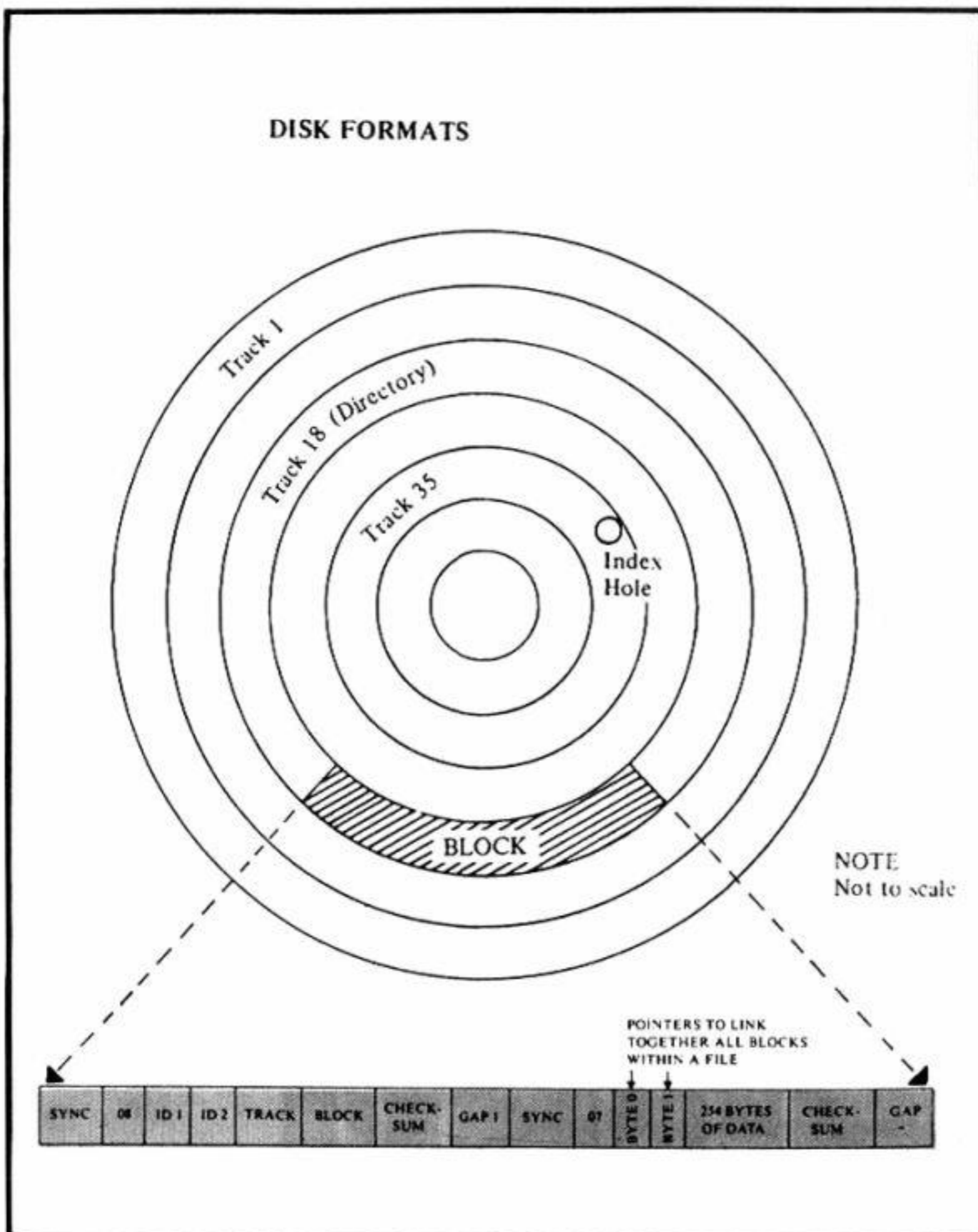
Se volete determinare con precisione il tempo impiegato dal VOSTRO drive, digitate il programmino di queste pagine, valido per il solo C/64. Se viene segnalato un tempo di posizionamento molto lungo (superiore a sette decimi di secondo), ciò sta ad indicare un disallineamento del disk drive, per cui sarà opportuno prendere misure adeguate (cioè far riallineare il drive).

Il programmino utilizza alcune particolarità: una routine di posizionamento del cursore del KERNAL viene chiamata per "disegnare" la nostra schermata e, dal momento che l'orologio TI$ è impreciso a causa dell'impiego di routine di I/O, è stato utilizzato il T.O.D. (time-of-day-clock) per calcolare i decimi di secondo impiegati dalla testina per posizionarsi sulla traccia desiderata. Il T.O.D. è esclusivo del C/64 e ne consegue che il listato non può essere adattato ad altri Commodore a meno di non ricorrere al linguaggio macchina e all'Interrupt.

Da test condotti su drive differenti si possono trovare valori medi oscillanti tra 4 e 6 decimi di secondo, mentre un drive che utilizza lo SPEED-DOS ha velocità di posizionamento compresa tra zero (cioè qualcosa meno di 1 decimo di secondo) e 3.

Nonostante dobbiate già sapere tutto su un disco, ricorderemo che esistono soltanto due modi per memorizzare dati su un dischetto. Il primo, detto "hard-sectoring" utilizza il forellino posto quasi al centro del disco.

Chissà quante volte vi sarete chiesti quale sia la sua funzione. Beh, nel sistema Commodore non ne ha alcuna, ma viene usato dai sistemi cosiddetti "hard-sectored" per indicare al Sistema Operativo del disco dove iniziano le varie tracce e settori. Vi sono anche certi sistemi che richiedono l'impiego di parecchi "fori" e, di con-

seguenza, di appositi dischetti.

Il Commodore impiega il sistema "soft-sectoring", vale a dire che, durante la FORMATTAZIONE del disco, vengono "scritti" dal SOFTware (donde il nome) sulla superficie del disco dei "segni" magnetici speciali che indicano al Sistema Operativo (DOS, acronimo di Disk Operating System) dove iniziano le tracce e i settori.

Così il Sistema Operativo ha modo di "controllare" che la testina venga posizionata proprio dove richiesto. Se comandiamo al disk drive di "caricare" (LOAD) un programma, la testina si posizionerà sulla traccia 18, che contiene la DIRECTORY, cercando tra i vari nomi dei programmi quello indicato, "leggendo" il numero della traccia e del settore iniziali del programma e poi, dopo aver spostato la testina sulla nuova traccia e settore, iniziando a "leggere" i dati dal disco.

Come certamente immaginate, non tutti i cerchi concentrici che rappresentano le TRACCE sono della stessa lunghezza. Perciò le tracce più distanti dal centro sono divise in un maggior numero di SETTORI delle tracce più vicine al centro del disco.

E' dunque importante sapere quanti siano i settori delle tracce dall'1 al 35:

TRACCIA	NUMERO DI SETTORI
1 -17	dallo 0 al 20
18-24	dallo 0 al 18
25-30	dallo 0 al 17
31-35	dallo 0 al 16

Dopo la cruda teoria passiamo a qualcosa di più dilettevole riguardante la formattazione.

Come sapete, la formattazione avviene mediante il comando:

OPEN 15,8,15,"N0:nome disco,id"

Orbene sostituendo i due valori dell'ID con due TOKEN del BASIC, la vostra directory assumerà un aspetto insolito. Per esempio il comando:

N0:nome disco,"+CHR$(146)+CHR$(128)

vi permetterà di leggere una testata di directory la quale, al posto delle consuete due lettere dell'ID, avrà i due comandi Basic WAIT ed END. Qui sotto troverete una lista dei tokens e dei relativi numeri decimali da utilizzare per ottenere l'effetto desiderato. Si potranno per esempio combinare i tokens per ottenere parole più o meno sensate, come RESTORE NEXT (140,130) o THEN VERIFY (167,149) oppure NOT FRE (168,184), eccetera.

Nro.dec.	Token
182	ABS
175	AND
198	ASC
193	ATN
199	CHR$
160	CLOSE
156	CLR
157	CMD
154	CONT
131	DATA
150	DEF
134	DIM
128	END
189	EXP
129	FOR
184	FRE
161	GET
141	GOSUB
137	GOTO

132	INPUT#
136	LET
200	LEFT$
155	LIST
147	LOAD
202	MID$
130	NEXT
168	NOT
159	OPEN
194	PEEK
151	POKE
153	PRINT
152	PRINT#
135	READ
140	RESTORE
142	RETURN
201	RIGHT$
187	RND
138	RUN
148	SAVE
180	SGN
191	SIN
166	SPC(
186	SQR
169	STEP
144	STOP
196	STR$
158	SYS
163	TAB(
167	THEN
183	USR
197	VAL
149	VERIFY
146	WAIT
190	COS
165	FN
139	IF
133	INPUT
181	INT
195	LEN
188	LOG
162	NEW
145	ON
176	OR
185	POS
143	REM
192	TAN

E ritorniamo alla nostra traccia 18. Benchè ogni settore di questa traccia sia importante, il settore zero è quello che ha il ruolo più delicato a causa dell'elevato numero di informazioni che possiede.

Con il programma "Monitor" pubblicato su CCC N.36 (o con un programma similare in vostro possesso) sarà ora possibile "sbirciare" in questo settore, intervenendo per modificare ciò che vogliamo.

Normalmente la traccia 18 settore 0 (esadec.: $12/$00) contiene le seguenti informazioni:

000-001
Linker: serve ad indicare quale sia il prossimo blocco di dati della directory. Di solito questo linker punta alla traccia 18, settore 1. Si noti che ogni blocco di dati del dischetto è fornito di un linker; se il valore è pari a $FF/$00 ciò sta ad indicare che il blocco in esame è l'ultimo della serie.

002
Indicatore di formato: nei dischetti formattati con il drive 1541 questo byte contiene sempre il valore di $41, corrispondente alla lettera "A". Se si cambia questo valore col monitor sarà impossibile poi scrivere qualcosa sul disco.

003
Non ha nessuna funzione

004-143
Contiene la B.A.M. (acronimo di Block Allocating Map) del disco. Ogni traccia è rappresentata da 4 byte. Il primo byte contiene il numero totale dei blocchi liberi di una traccia e gli altri 3 byte (per un totale quindi di 24 bit) indicano quali settori di quella data traccia siano quelli liberi (bit=1) e quali quelli occupati (bit=0).

144-161
Questi byte contengono il nome del disco. Gli spazi vuoti vengono riempiti con SHIFT-ed SPACEs ($A0, decimale 160).

162-163
ID (identificatore) del disco.

164
Contiene il valore $A0

165-166
Indicatore di formato per la testata della directory. Di solito contiene i valori di $32/$41 che vengono visualizzati come "2A".

167-170
Contengono il valore $A0

171-255
Solitamente questo spazio non viene utilizzato.

Utilizzate il monitor per studiare il contenuto della traccia 18 e del settore 0 dei vostri dischi, ma fate attenzione che qualunque vostro intervento con l'opzione "W" potrebbe avere effetti letali sui dati e programmi in esso contenuti.

Il secondo programma visualizza sullo schermo il contenuto della BAM oppure, a scelta, controlla se i singoli settori di un disco presentano problemi. Si consiglia di compilare il programma a causa della lentezza del Basic. Gli utenti di altri computer, dopo un attento studio del listato, possono adattarlo ai propri calcolatori.

```
100 REM DETERMINAZIONE VELOCITA
    ,
110 REM DRIVE 1541, 1571
120 REM BY PAOLO AGOSTINI
140 REM SOLO PER C/64 A CAUSA D
    EL RICORSO ALL'OROLOGIO "SP
    ECIALE" INTERNO
150 :
160 PRINTCHR$(147):POKE 53280,0
    :POKE 53281,0:GOSUB 410:GOT
    O 200
170 REM POSIZIONAMENTO CURSORE
```

```
180 POKE 214,R%:POKE 211,C%:SYS
    58640:RETURN
190 REM LOOP PRINCIPALE
200 OPEN 8,8,8,"#"
210 T=1
220 TR=T(T):GOSUB 350:PRINT#15,
    "U1 8 0 ";TR;1
230 GOSUB 380:R%=1:C%=0:GOSUB 1
    80:PRINTCHR$(18);CHR$(30);L
    EFT$(SP$,13);
240 PRINTCHR$(158);LEFT$(SP$,13
    );CHR$(28);LEFT$(SP$,13)
250 R%=1:C%=0:GOSUB 180:IF Z<10
     THEN PRINT TAB(Z*4);CHR$(5
    );CHR$(221);CHR$(154)
260 R%=TR+5:C%=2:IF TR>18 THEN
    R%=TR-13:C%=22
270 GOSUB 180:TR$=STR$(TR):TR$=
    RIGHT$(TR$,LEN(TR$)-1):TR$=
    RIGHT$("00"+TR$,2)
280 PRINTCHR$(18);"TRACCIA ";TR
    $;":";Z
290 R%=TR+5:C%=2:IF TR>18 THEN
    R%=TR-13:C%=22
300 GOSUB 180:IF Z<7 THEN PRINT
    "TRACCIA ";TR$;":";Z
310 GET A$:IF A$<>"" THEN CLOSE
     8:CLOSE 15:END
320 T=T+1:IF T=36 THEN 210
330 GOTO 220
340 REM AZZERA OROLOGIO INTERNO
350 POKE 56334,129:FOR I=56331
    TO 56328 STEP -1
360 POKE I,0:NEXT:RETURN
370 REM LETTURA SEC. E DECIMI D
    I SECONDO
380 X=PEEK(56329):Z1=0:IF X<>0
    THEN Z1=(X/16)*10+(X  AND
    15):Z1=Z1*10
390 X=PEEK(56328):Z=(X  AND  15
    )+Z1:RETURN
400 REM INIZIALIZZAZIONE
410 DIM T(35):FOR I=1 TO 35:REA
    D T(I):NEXT
420 DATA 1,19,2,20,3,21,4,22,5,
    23,6,24,7,25,8,26,9,27
430 DATA 10,28,11,29,12,30,13,3
    1,14,32,15,33,16,34,17,35,1
    8
440 FOR I=1 TO 39:SP$=SP$+CHR$(
    32):NEXT
450 OPEN 15,8,15,"I0":INPUT#15,
    E1,E1$
460 IF E1 THEN PRINT"ERRORE DIS
    CO:"E1;E1$:CLOSE 15:END
470 R%=0:C%=4:GOSUB 180:PRINT"A
    LTA":C%=16:GOSUB 180:PRINT"
    NORMALE"
480 C%=29:GOSUB 180:PRINT"BASSA
    "
490 R%=2:C%=2:GOSUB 180:PRINT"V
    ELOCITA' DI POSIZIONAMENTO
    TESTINA"
500 R%=3:C%=5:GOSUB 180:PRINT"E
    SPRESSA IN DECIMI DI SECOND
    O"
510 R%=4:C%=12:GOSUB 170:PRINT"
    SPACE PER FINIRE"
520 RETURN
530 END
```

```
 10 REM CHECK DISK ED ESAME B.A
    .M.
 20 REM BY PAOLO AGOSTINI
 30 :
100 GOSUB 1500
110 PRINTCHR$(147);P$; TAB(11);
    "INSERIRE UN DISCO E"
120 PRINT TAB(12);"PREMERE UN T
    ASTO.":GOSUB 150:GOTO 580
130 :
140 REM ATTENDE PRESSIONE DI UN
     TASTO
150 POKE SID+4,21:FOR I=1 TO 30
    0:GET A$:IF A$<>"" THEN 190
160 NEXT
170 POKE SID+4,20:FOR I=1 TO 10
    0:GET A$:IF A$<>"" THEN 190
180 NEXT:GOTO 150
190 RETURN
200 :
210 REM CHECK ERROR CHANNEL
220 INPUT#15,E1,E1$,E2,E3
230 RETURN
240 :
250 REM SCREEN FORMATTING
260 IN=1146:PRINTCHR$(147): TAB
    (12);"1"; TAB(22);"2"; TAB(
    32);"3"
270 FOR I=1 TO 35:PRINT TAB(I+2
    );RIGHT$(STR$(I),1);:NEXT:P
    RINT
```

```
280 PRINT TAB(2);CHR$(176);:FOR
     I=1 TO 35:PRINT TAB(I+2);C
    HR$(192);:NEXT
290 PRINTCHR$(174)
300 FOR I=0 TO 20: I$=RIGHT$(ST
    R$(I),LEN(STR$(I))-1)+CHR$(
    221)
310 PRINT TAB(3-LEN(I$));I$; TA
    B(38);CHR$(221):NEXT
320 BV$="[RVS] [RVOFF]":FOR S=1
    7 TO 20:IF S=17 THEN T=31
330 IF S=18 THEN T=25
340 IF S=19 OR S=20 THEN T=18
350 PRINTCHR$(19);:FOR I=-2 TO
    S:PRINT"[DOWN]";:NEXT
360 FOR J=T TO 35:PRINT TAB(J+2
    );BV$;:NEXT:PRINT:NEXT
370 BV$=BV$(1):P=1983:GOSUB 430
380 BV$="TRACK:":P=1890:GOSUB 4
    30
390 BV$="SECTOR:":P=1929:GOSUB
    430
400 PRINTCHR$(19);:RETURN
410 :
420 REM REVERSE ONTO SCREEN
430 FOR I=1 TO LEN(BV$)
440 Z=ASC(MID$(BV$,I,1)):IF Z<3
    2 THEN POKE P+I,160:GOTO 48
    0
450 IF Z<64 THEN POKE P+I,Z+128
460 IF Z>64 THEN POKE P+I,Z+64
470 POKE P+I+54272,PEEK(646)
480 NEXT:RETURN
490 :
500 REM PRINT TRACK & SECTOR
510 BV$=STR$(T)+" ":P=1896:GOSU
    B 430
520 BV$=STR$(S)+" ":P=1936:GOSU
    B 430
530 POKE IN+(S*40)+T,CAR
540 POKE IN+(S*40)+T+54272,PEEK
    (646)
550 RETURN
560 :
570 REM MENU PRINCIPALE
580 GOSUB 260:PRINTP$; TAB(18)"
    MENU":PRINT
590 PRINT:PRINT TAB(7);"[RVS] 1
     [RVOFF]  CONTROLLO DISCO"
600 PRINT:PRINT TAB(7);"[RVS] 2
     [RVOFF]  VISUALIZZAZIONE B
    .A.M."
610 PRINT:PRINT:PRINT TAB(17):"
    QUALE ?"
620 GOSUB 150:IF A$<>"1" AND A$
    <>"2" THEN 620
630 IF A$="1" THEN 910
640 :
650 REM VISUALIZZAZIONE B.A.M.
660 GOSUB 1170:REM INIZIALIZZAZ
    IONE
670 BV$=BV$(2):P=1983:GOSUB 430
680 PRINT#15,"U1 2 0 18 0"
690 PRINT#15,"B-P 2 4"
700 BV$="BLOCKS FREE:":P=1964:G
    OSUB 430
710 FOR T=1 TO 35
720 GET #2,SC$:SC$=SC$+CHR$(0)
730 SC=ASC(SC$):IF T<>18 THEN T
    S=TS+SC
740 BV$=STR$(TS):P=1976:GOSUB 4
    30
750 FOR H=0 TO 2:GET #2,A$:IF A
    $="" THEN A$=CHR$(0)
760 SB(H)=ASC(A$):NEXT
770 IF T<36 THEN SA=16
780 IF T<31 THEN SA=17
790 IF T<25 THEN SA=18
800 IF T<18 THEN SA=20
810 FOR S=0 TO SA
820 CAR=58:IF  FNS(S)=0  THEN C
    AR=42
830 GOSUB 510
840 NEXTS:NEXTT
850 CLOSE 2:CLOSE 15
860 BV$=BV$(3):P=1983:GOSUB 430
870 GOSUB 150:IF A$="←" THEN GO
    SUB 1260
880 GOTO 580
890 :
900 REM CONTROLLO DISCO
910 GOSUB 1170:REM INIZIALIZZAZ
    IONE
920 BV$=BV$(4):P=1983:GOSUB 430
930 BV$="STATUS: 0":P=1969:GOSU
    B 430
940 FOR T=1 TO 35
950 IF T<36 THEN SA=16
960 IF T<31 THEN SA=17
970 IF T<25 THEN SA=18
```

```
 980 IF T<18 THEN SA=20
 990 FOR S=0 TO SA
1000 PRINT#15,"U1 2 0 ";T;S
1010 GOSUB 220:IF E1=0 THEN CAR=
     45:GOSUB 510 :REM CAR=CARAT
     TERE VISUALIZZATO
1020 IF E1<>0 THEN CAR=5:GOSUB 5
     10:GOSUB 1100
1030 NEXTS:NEXTT
1040 CLOSE 2:CLOSE 15
1050 BV$=BV$(5):P=1983:GOSUB 430
1060 GOSUB 150:IF A$="←" THEN GO
     SUB 1260
1070 GOTO 580
1080 :
1090 REM ROUTINE DI ERRORE
1100 BV$=STR$(E1):P=1976:GOSUB 4
     30
1110 CLOSE 2:CLOSE 15:OPEN 15,8,
     15,"I0":GOSUB 220
1120 IF E1 THEN BV$=BV$(7):P=198
     3:GOSUB 430:END
1130 OPEN 2,8,2,"#":GOSUB 220
1140 BV$=STR$(E1)+" ":P=1976:GOS
     UB 430:RETURN
1150 :
1160 REM INIZIALIZZAZIONE CANALI
1170 GOSUB 260:OPEN 15,8,15,"I0"
     :OPEN 2,8,2,"#":TS=0
1180 GOSUB 220:IF E1=0 THEN RETU
     RN
1190 PRINTP$; TAB(14);"ERRORE DI
     SCO"
1200 ER$=STR$(E1)+","+E1$+","+ST
     R$(E2)+","+STR$(E3)
1210 PRINT TAB((40-LEN(ER$))/2);
     ER$
1220 PRINT:PRINT TAB(12);"PREMER
     E UN TASTO":GOSUB 150
1230 CLOSE 2:CLOSE 15:GOTO 580
1240 :
1250 REM STAMPA SCHERMO
1260 OPEN 127,4:POKE 768,185:PRI
     NT#127
1270 CLOSE 127:POKE 768,139
1280 IF ST<>-128 THEN 1310
1290 BV$=BV$(6):P=1983:GOSUB 430
1300 GOSUB 150:GOTO 1260
1305 REM PRENDE NOME DISCO
1310 OPEN 15,8,15,"I0":OPEN 2,8,
     2,"#"
1320 PRINT#15,"U1 2 0 18 0"
1330 PRINT#15,"B-P 2 144"
1340 FOR I=0 TO 16:GET #2,A$:IF
     A$="" THEN A$=CHR$(0)
1350 IF A$=CHR$(160) THEN 1370
1360 ND$=ND$+A$
1370 NEXT
1380 CLOSE 2:CLOSE 15:PRINT"[HOM
     E]";
1390 IF LEN(ND$)<40 THEN ND$=CHR
     $(32)+ND$
1400 IF LEN(ND$)<40 THEN ND$=ND$
     +CHR$(32):GOTO 1390
1410 BV$=ND$:P=1983:GOSUB 430
1420 OPEN 1,3:OPEN 4,4
1430 FOR I=1 TO 25:FOR J=1 TO 40
1440 GET #1,A$:B$=B$+A$:NEXTJ
1450 PRINT#4,B$;:REM EVENTUALMEN
     TE ELIMINARE IL PUNTO E VIR
     GOLA
1460 B$="":NEXTI
1470 CLOSE 4:CLOSE 1:GOTO 580
1480 :
1490 REM INIZIALIZZAZIONE
1500 CAR=42:T=1:S=0:P$="[HOME][7
      DOWN]":DIM BV$(15)
1510 DEF  FNS(Z)=2↑(S-INT(S/8)*8
     ) AND (SB(INT(S/8)))
1520 SID=54272:FOR H=SID TO SID+
     24:POKE H,0:NEXT
1530 POKE SID+1,130:POKE SID+5,9
     :POKE SID+15,30:POKE SID+24
     ,15:POKE SID+4,20
1540 BV$(1)="** PAOLO AGOSTINI -
      DISK CHECKER 1986 *"
1550 BV$(2)="  BLOCK AVAILABILIT
     Y MAP - ATTENDERE    "
1560 BV$(3)="B.A.M. -PREMI UN TA
     STO (← PER STAMPARE)"
1570 BV$(4)="   CONTROLLO DISCO
     - ATTENDERE PREGO    "
1580 BV$(5)="CHECKUP-PREMI UN TA
     STO (← PER STAMPARE)"
1590 BV$(6)=" ACCENDERE LA STAMP
     ANTE PER STAMPARE !  "
1600 BV$(7)=" ERRORE DI INIZIALI
     ZZAZIONE DISCO !!!!  "
1610 RETURN
1620 END
```

Biglia elettronica

Un gioco di totale disimpegno che metterà duramente alla prova riflessi e capacità di osservazione

di Maurizio Dell'Abate

Dopo aver risposto alla domanda sul computer adoperato, è necessario scegliere il livello di difficoltà del gioco, che può oscillare tra uno e dieci.

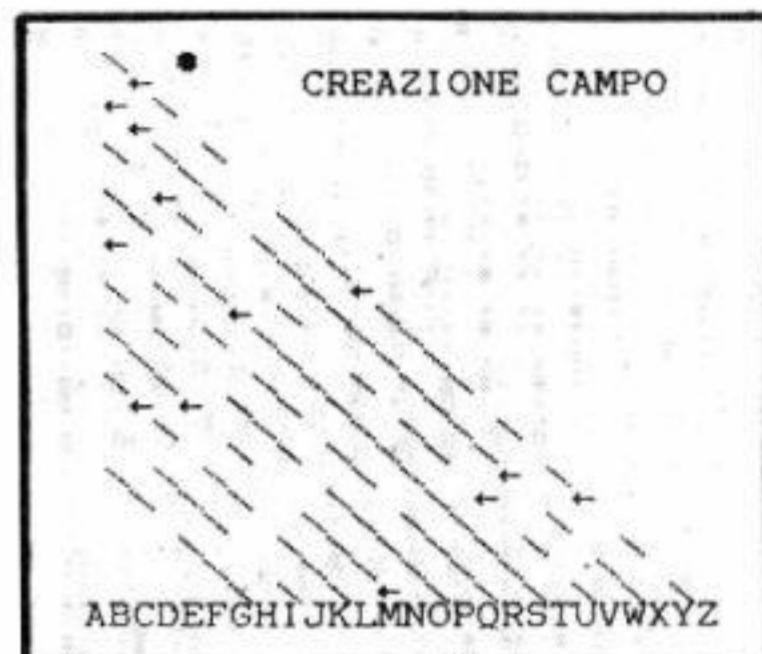
Sul video saranno tracciate alcune linee oblique (ottenute mediante caratteri grafici) interrotte in più punti; sopra di esse è rappresentata una pallina.

Scopo del gioco è prevedere dove andrà a finire la biglia quando verrà, in seguito, lasciata cadere dal punto in cui è visualizzata.

La biglia, com'è intuitivo, segue un percorso verticale se non trova ostacoli; se invece incontra un ostacolo (linea) prosegue la sua caduta in diagonale fino a che non incontra un "buco".

Il discorso potrebbe sembrare complesso, ma comprendere il gioco è incredibilmente semplice. Nei livelli di difficoltà superiori al primo, sono presenti sul video, oltre alle sbarrette diagonali, anche alcune "freccette" a sinistra: se la biglia cade su una di queste, si sposta immediatamente a sinistra di una posizione.

La biglia giungerà, prima o poi, sull'ultima riga di schermo su cui sono visualizzate le lettere dell'alfabeto. Prima che il tempo messo a disposizione (inversamente proporzionale alla difficoltà) si esaurisca, è opportuno comunicare al programma la lettera su cui si prevede che cadrà la biglia.

Dopo aver premuto il tasto corrispondente alla lettera la pallina inizia la simulazione della discesa; viene quindi assegnato un punteggio, legato al livello di difficoltà, che tiene conto anche dell'entità di un eventuale errore di previsione.

Effettuati dieci tentativi il gioco termina e viene reso noto il punteggio.

Provate ad individuare il percorso esatto nell'esempio riportato in figura in cui la biglia raggiunge il 12imo carattere.

SCHEDA TECNICA

Videogame a scopo didattico valido per computer C/64, C/16, Plus/4, ma facilmente adattabile ad altri computer Commodore.
Programma consigliato ai lettori principianti
Anche il programma pubblicato in queste pagine è contenuto nel disco "Directory" di questo mese.

```
100 REM  BIGLIA ELETTRONICA
110 REM  MICRO-GAME BY M. DELL'
    ABATE
120 REM  PER C/64-128-16 E PLUS
    /4
130 :
140 PRINTCHR$(147)
150 PRINT"QUALE COMPUTER STAI U
    SANDO?"
160 PRINT:PRINT"1. COMMODORE 64
    "
170 PRINT"2. COMMODORE 16 - PLU
    S/4"
180 GET H$
190 IF H$="1" THEN A=1024:GOTO
    220
200 IF H$="2" THEN A=3072:GOTO
    220
210 GOTO 180
220 PRINTCHR$(147)
230 INPUT " LIVELLO DI DIFFICOL
    TA' DA 0 A 10";LV
240 IF LV<0 OR LV>10 THEN  RUN:
    REM  VALORI FUORI DAI LIMIT
    I CONSENTITI
250 GOTO 270
260 PRINTCHR$(19); TAB(18);CHR$
    (17);A$:RETURN
270 SC=0
280 B=10:REM  NUMERO DIAGONALI
290 :
300 REM  *** CREAZIONE CAMPO **
    *
310 :
```

```
320 FOR DL=1 TO 10:REM  NUMERO
    TENTATIVI
330 PRINTCHR$(147):FOR G=0 TO 2
    3:PRINTCHR$(17);:NEXT:REM
    CURSORE SULL'ULTIMA RIGA
340 PRINT TAB(9);"ABCDEFGHIJKLM
    NOPQRSTUVWXYZ";
350 A$="CREAZIONE CAMPO       "
360 GOSUB 260
370 POKE A+13,81:REM  CODICE PA
    LLINA
380 FOR C=1 TO B
390 CT=A+80*(C-1)+10
400 CS=77:REM  CODICE BARRETTA
    DIAG.
410 IF RND(0)<.2 THEN CS=32:REM
      COD. SPC
420 IF RND(0)<LV/25 THEN CS=32
430 IF RND(0)<LV/70 THEN CS=31:
    REM  COD. ←
440 POKE CT,CS
450 CT=CT+41
460 IF CT>A+959 THEN 480
470 GOTO 400
480 NEXT
490 :
500 REM  *** INPUT LETTERA ***
510 REM  *** E CONTEGGIO   ***
520 :
530 D$="                    ":R
    EM  20 SPC
540 FOR CI=(20-LV)*7 TO 0 STEP
    -1:REM  CONTEGGIO IN RAPPOR
    TO AL LIVELLO DI DIFF.
550 A$=STR$(CI)+LEFT$(D$,20-LEN
    (STR$(CI))):GOSUB 260
560 GET W$:IF W$<"A" OR W$>"Z"
    THEN NEXT:REM  CONTROLLA LA
    TASTIERA
570 A$="HAI SCELTO: "+W$+"
      ":GOSUB 260:REM  7 SPAZI
580 FOR CI=0 TO 999:NEXT:REM  R
    ITARDO
590 :
600 REM  *** DISCESA PALLINA **
    *
610 :
620 A$="DISCESA PALLINA...   ":G
    OSUB 260
630 XP=13:YP=0:Q=A
640 LC=A+XP+YP*40:IF LC>A+959 T
    HEN 760
650 POKE LC,81:REM  CODICE PALL
    INA
660 POKE Q,32:Q=LC:SO=PEEK(LC+4
    0)
670 IF SO=77 THEN XP=XP+1:YP=YP
    +1:GOTO 710:REM  PALLINA SU
     UNA BARRETTA
680 IF SO=32 THEN YP=YP+1:GOTO
    710:REM  LA PALLINA CADE NE
    L VUOTO
690 IF SO=31 THEN XP=XP-1:YP=YP
    +1:GOTO 710:REM  PALLINA SU
    LLA FRECCETTA ("←")
700 GOTO 760
710 FOR TE=0 TO 50:NEXT:REM  RI
    TARDO
720 GOTO 640
730 :
740 REM  *** CONTROLLO E PUNTEG
    GIO ***
750 :
760 FOR MA=0 TO 50:NEXT:CH=PEEK
    (LC+40)+64:POKE LC,32:POKE
    LC+40,81:REM  CODE BIGLIA
770 IF W$="" THEN SP=0:GOTO 810
    :REM  NESSUNA LETTERA E' ST
    ATA SELEZIONATA
780 IF ASC(W$)=CH THEN OP=60:A$
    ="PERFETTO!!          ":GOS
    UB 260:GOTO 800:REM  10 SPC
790 OP=30-ABS(ASC(W$)-CH):A$=D$
    :GOSUB 260
800 SP=OP*(LV+1)
810 SC=SC+SP:REM  INCREMENTO PU
    NTEGGIO
820 FOR CI=0 TO 1999:NEXT:REM
    DUE SECONDI
830 PRINTCHR$(147):PRINT" BONUS
    :";SP
840 FOR CI=0 TO 999:NEXT
850 NEXT
860 PRINTCHR$(147)
870 PRINT"PUNTEGGIO";SC:PRINT:P
    RINT
880 PRINT"PREMI UN TASTO"
890 GET A$:IF A$="" THEN 890
900 GOTO 220:REM  NUOVA PARTITA
910 END
```

Grafica in Basic 3.5 e 7.0

Due routine per disegni casuali (e non) che girano allo stesso modo su C/16, Plus-4 e C/128. E inoltre: una routine di hard copy per la pagina grafica del C-128!

di Mauro e Stefano Ciurli

Il programma "Random project" genera figure che potremo definire frattali (o tridimensionali) mediante l'inserimento di ben cinque variabili che, nel listato, sono chiamate:

R0 : primo raggio
R1 : secondo raggio
R2 : terzo raggio
NG : numero di giri
PA : passo

Queste vengono memorizzate e disegnate, in seguito, tramite il comando Draw, formando figure simmetriche che forniscono, quasi sempre, l'illusione della terza dimensione.

La routine di calcolo, cuore del programma, può essere eseguita in modalità Fast (solo dai possessori di C/128) mentre i vari tratti disegnati possono esser riprodotti in modalità Slow in modo da osservare l'evolversi del disegno.

Naturalmente il listato pubblicato è privo di tali istruzioni per consentire la perfetta compatibilità con il C/16 e il Plus-4.

Nel menù iniziale vengono inoltre forniti alcuni valori già sperimentati, nonchè la possibilità di ripetere l'inserimento.

Il "numero giri" permette di definire il numero dei passaggi che la figura deve compiere, mentre per "passo" si intende l'incremento tra i passaggi stessi.

Chi possiede il C/128 può digitare, in coda, la parte del secondo listato contenente i Data relativi alla routine di hard copy (che, lo ricordiamo, funziona solo per il C/128).

Il programma Quark

Il programma Quark permette di generare figure complesse nella modalità grafica 1 tramite valori Random e mediante l'istruzione Circle.

La figura che si forma sullo schermo può essere fermata, e fatta ripartire, mediante la pressione della barra spaziatrice, mentre premendo il tasto asterisco (*) si ritorna al menù.

Le opzioni previste sono le seguenti:

1: rivedi quark.
Permette di ritornare allo schermo grafico e poi di nuovo al menù.

2: hardcopy.
Stampa il disegno ottenuto, grazie alla routine in linguaggio macchina, su Mps 801/803 o compatibili.

3: stampa variabili.
Stampa le variabili pseudocasuali trovate, sia sullo schermo sia su stampante (se è accesa) permettendo di prenderne nota in modo da riprodurre il disegno, in un proprio programma, con la semplice istruzione Circle.

4: centratura.
Permette di centrare il disegno eseguito in un'altra parte dello schermo, consentendo di ottimizzarne il posizionamento.

5: continua drawing.
Consente di iniziare il processo costruendo una figura.

6: scelta variabili.
Permette di inserire le variabili desiderate nell'istruzione circle, offrendo, così, la possibilità di generare figure ben precise (tra cui quelle realizzate in precedenza) o di modificare figure pseudocasuali create dallo stesso computer.

La routine di calcolo, cuore del programma, viene eseguita in fast mode; i valori trovati vengono, naturalmente, compressi, per evitare che la figura possa trovarsi al di fuori dello schermo. Ad ogni modo le istruzioni Trap e Resume evitano il verificarsi di errori tra cui, tra i più probabili, quelli

Le variabili elaborate sono:

Xi-Yi: sono valori trovati per aggregazione tramite le altre variabili X, Y e Z, permettendo l'ottimizzazione della centratura e una casualità maggiore.

RT: fornisce la base di partenza del ciclo di disegno, che arriva a 360.

R-R1: permettono di assegnare forme ellissoidali alla figura.

H-S: forniscono la rotazione in gradi e i gradi tra segmenti.

Alcuni suggerimenti

Il programma Random Project è digitabile perfino sul C/16 senza apportare alcuna modifica. Per "Quark", invece, è possibile digitarlo anche sul piccolo computer a patto di escludere la routine in linguaggio macchina (tutte le righe Data e quelle che la gestiscono) e tutte le istruzioni che non hanno significato nel Basic 3.0.

Anche le Rem devono essere omesse, mentre le righe relative al menu, "ricche" di frasi esplicative, devono essere ridotte all'osso per evitare i messaggi di "Out of memory error".

Si ricorda che, in fase di Input, è necessario digitare solo i parametri che si intendono cambiare: gli altri, infatti, rimangono inalterati anche se ci si limita a battere il solo tasto Return.

Per ciò che riguarda la routine di Hard-Copy che, ripetiamo, gira solo su C/128 in modo 128, consigliamo di digitarla con la massima attenzione per evitare il Crash del sistema.

SCHEDA TECNICA

Software applicativo per:

grafica
didattica
stampante

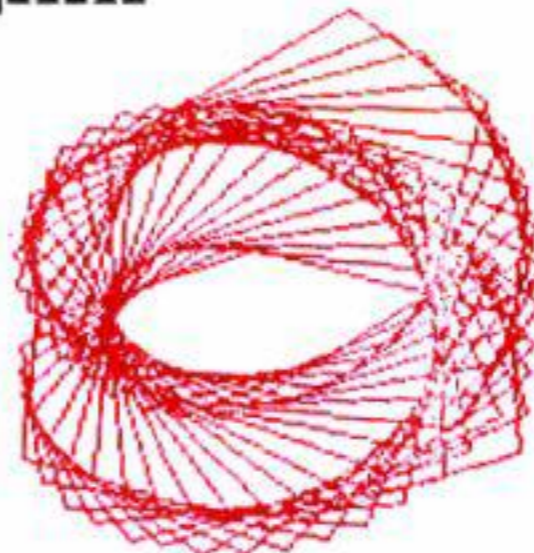
Anche i programmi pubblicati in queste pagine sono contenuti nel disco "Directory" di questo mese.

Idoneo per computer: C/16, Plus/4, C/128

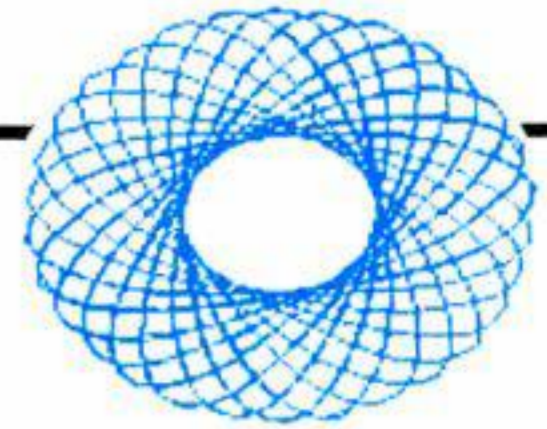
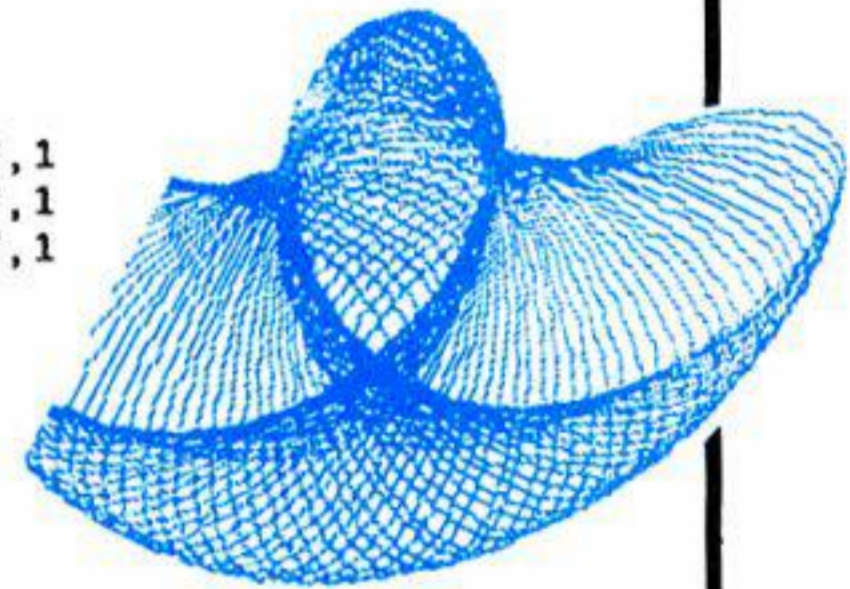
Difficilmente adattabile ad altri computer Commodore

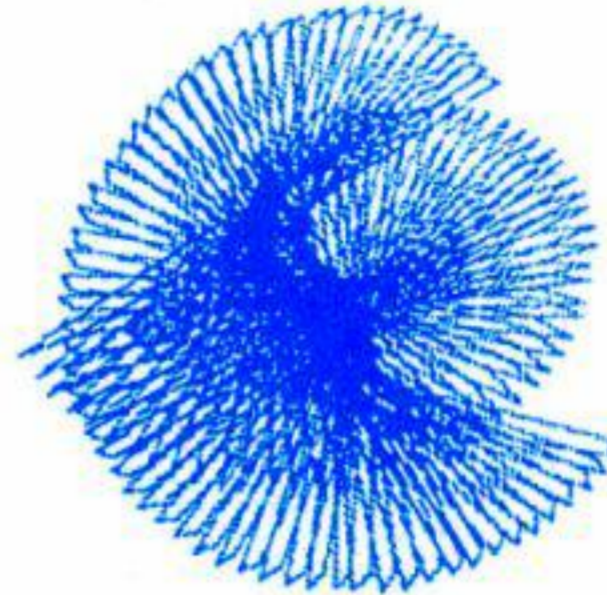
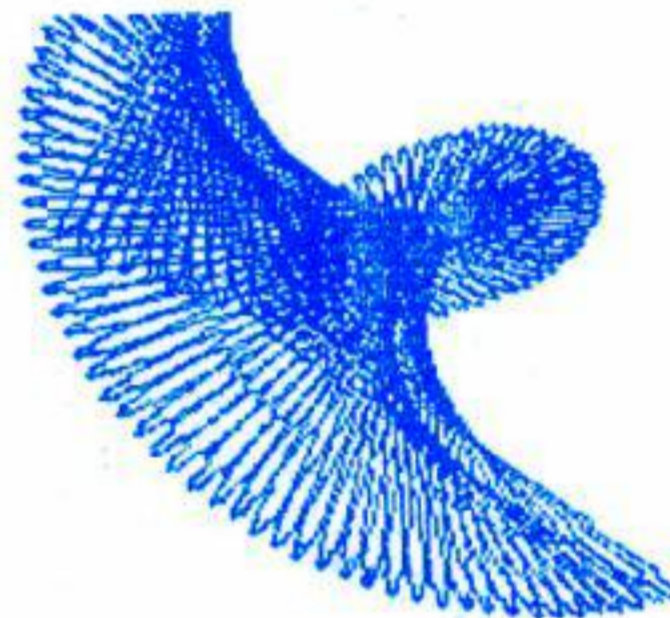
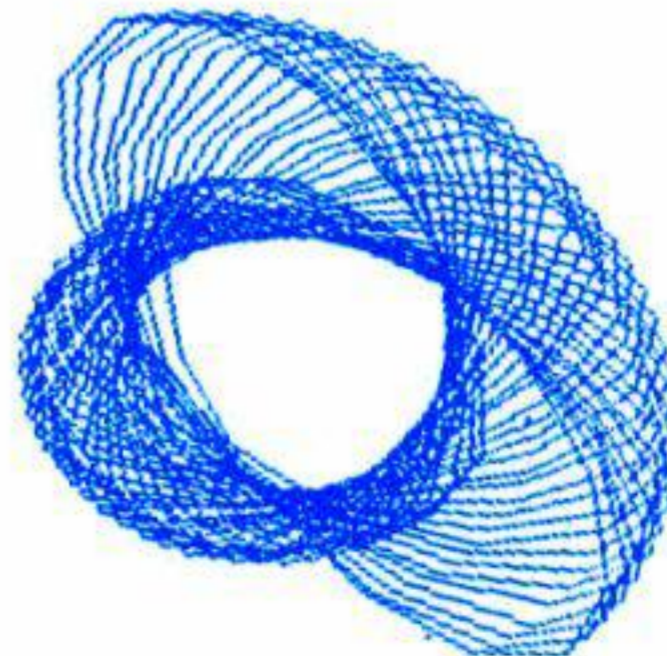
Consigliato l'uso della stampante Mps-803 (solo per C/128)

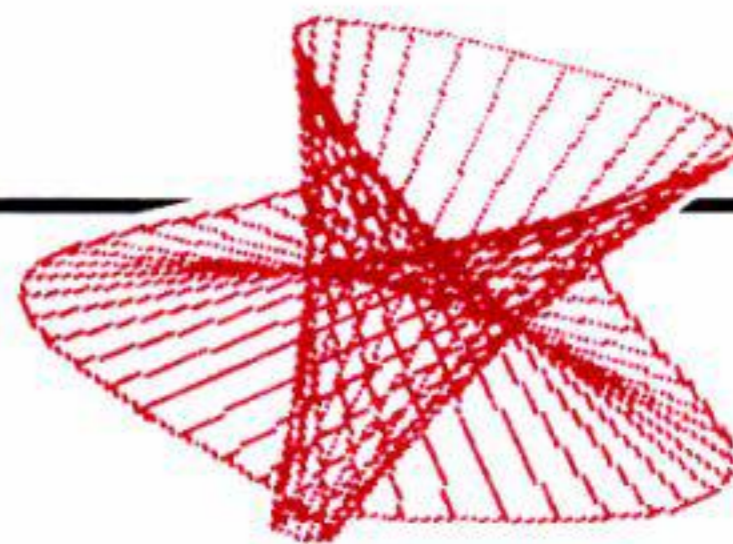
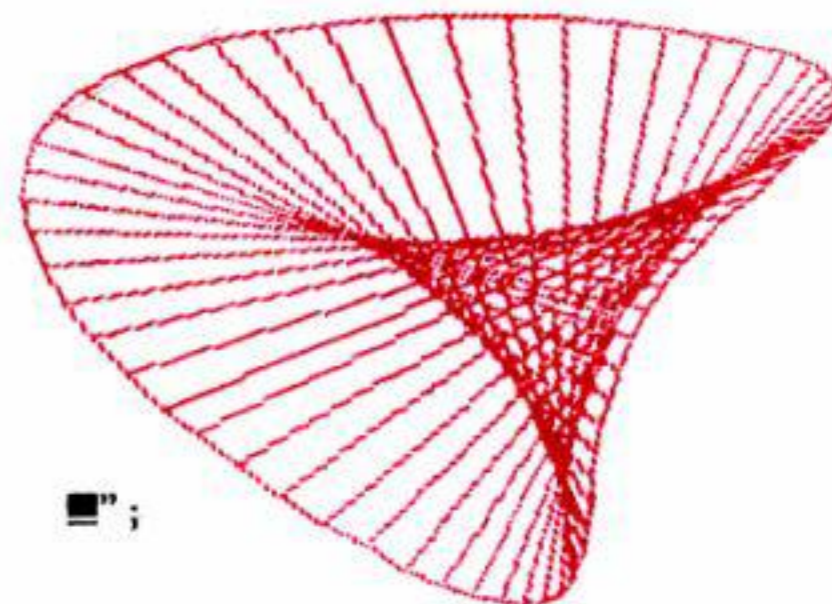
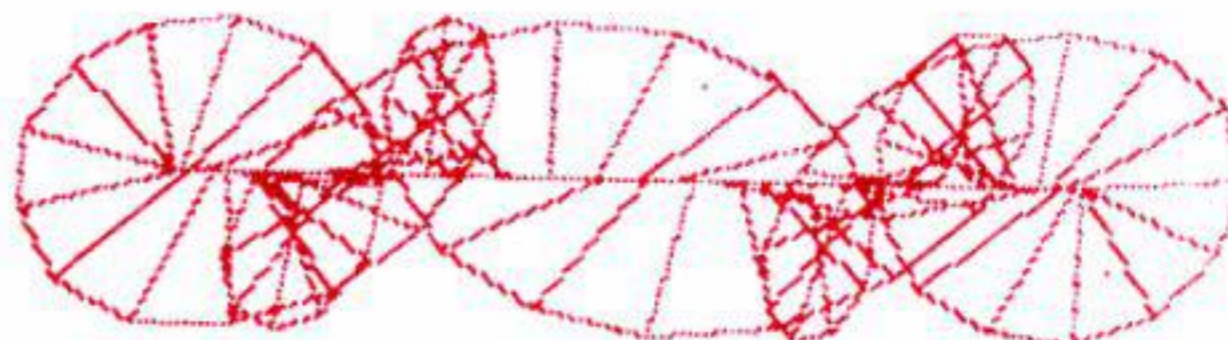
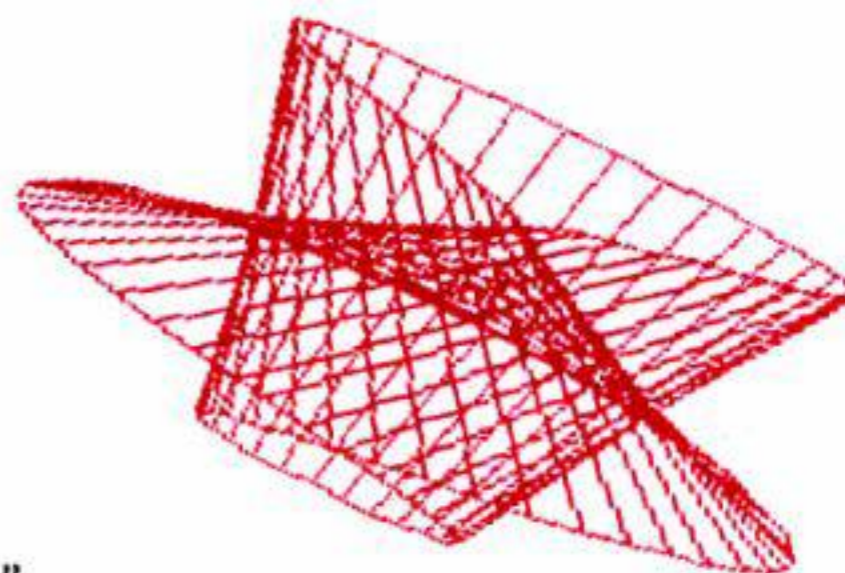
Consigliato a tutti gli appassionati di grafica

```
100 REM     Q U A R K          C / 128
110 :
120 REM GRAFICA PER C/128 IN MODO 128
130 REM IN MODALITA' 40 COLONNE
140 :
150 REM CONTIENE ROUTINE L.M. PER HARD COPY DELLA PAGINA GRAFICA
160 REM SU STAMPANTI MPS/803 E COMPATIBILI
170 :
180 REM BY MAURO E STEFANO CIURLI DI ROMA -
190 :
200 TRAP1540
210 COLOR0,1:COLOR4,1:COLOR1,8:GRAPHIC1,1
220 FORI=1TO30:X=INT(RND(1)*320)+1:Y=INT(RND(1)*170)+1:DRAW1,X,Y:NEXT
230 SOUND1,48000,190,1,0,100,1,0
240 FORI=10TO190STEP10:DRAW,10,ITOI,190
250 DRAW,310,ITO320-I,190:NEXT
260 SOUND2,4000,50,0,2000,400
270 SLEEP2
280 SCNCLR
290 GOSUB1060
300 COLOR0,1:COLOR4,1:COLOR1,8
310 CHAR1,4,12,"SBARRA=STOP/CONT DISEGNO *=MENU'",1:SLEEP2:SCNCLR1
320 GRAPHIC1:FAST
330 REM ---------ROUTINE DI CALCOLO-----
340 X=INT(RND(0)*161)+79
350 Y=INT(RND(0)*101)+49
360 R=INT(RND(0)*62)+15
370 IFX-R<42THENX=X+1:GOTO370
380 IFX+R>270THENX=X-1:GOTO380
390 R1=INT(RND(0)*77)
400 IFY-R1<39THENY=Y+1:GOTO400
410 IFY+R1>162THENY=Y-1:GOTO410
420 RT=INT(RND(0)*181)
430 P=INT(RND(0)*18)+1
440 Z=INT(RND(0)*7)-3
450 Z1=INT(RND(0)*7)-3
```

X= 145 Y= 83 RT= 119 P= 9

R= 69 R1= 42 S= 61 Z= 0 Z1= 2

```
460 S=INT(RND(0)*181)
470 ZZ=RND((0)*1)
480 XI=X:YI=Y:FORH=RTTO360STEPP
490 IFZZ>.82THEN520
500 XI=XI+Z:YI=YI+Z1:IFXI<5ORXI>314ORYI<5ORYI>194THEN590
510 IFZZ<.31THEN530
520 IFH  >360THENH=360
530 SLOW
540 CIRCLE1,XI,YI,R,R1,0,360,H,S
550 GETU$:IFU$=" "THENGOSUB1020
560 IFU$="*"THEN610
570 NEXTH
580 IFRI=1THENRI=0:CHAR1,19,24,"PREMI '*' PER USCIRE..":GETKEYB$:GOTO610
590 GETKEYW$
600 REM ---------MENU' OPZIONI----------
610 GRAPHIC0:COLOR0,1:COLOR4,1:COLOR5,2:SCNCLR
620 CHAR1,0,0,"                                        ",1
630 CHAR1,0,1," QUARK   QUARK   QUARK   QUARK   QUARK  ",1
640 CHAR1,0,2,"                                        ",1
650 CHAR1,10,5,"                         ",1
660 CHAR1,10,6," * 1 RIVEDI QUARK        ",1
670 CHAR1,10,7," * 2 HARDCOPY            ",1
680 CHAR1,10,8," * 3 STAMPA VARIABILI    ",1
690 CHAR1,10,9," * 4 CORREGGI CENTRATURA ",1
700 CHAR1,10,10," * 5 CONTINUA DRAWING    ",1
710 CHAR1,10,11," * 6 DRAW QUARK          ",1
720 CHAR1,10,12,"                         ",1
730 CHAR1,0,24,"    SPACE-RUN-STOP DRAW....*-MENU'     ",1
740 GETKEYA$
750 IFA$="1"THENGRAPHIC1:RI=1:GOTO580
760 IFA$="2"THENSYS5392
770 IFA$="3"THENBEGIN:CHAR1,11,14,"LISTA VARIABILI"
780 PRINT:PRINT:PRINTTAB(8)"X=";X:PRINTTAB(15)"]Y=";Y:PRINTTAB(22)"]RT=";RT:PRIN
TTAB(8)"P=";P:PRINTTAB(15)"]R=";R:PRINTTAB(22)"]R1=";R1
790 PRINTTAB(8)"S=";S:PRINTTAB(15)"]Z=";Z:PRINTTAB(22)"]Z1=";Z1
800 OPEN4,4:CLOSE4:IFST<0THEN820
810 OPEN4,4:PRINT#4,"LISTA VARIABILI":PRINT#4,CHR$(13),"X=";X;"Y=";Y;"RT=";RT;"P
=";P;"R=";R;"R1=";R1;"S=";S;"Z=";Z;"Z1=";Z1
820 CLOSE4:BEND
830 IFA$="5"THENSCNCLR1:GRAPHIC1:GOTO320
840 IFA$="4"THENBEGIN:WINDOW0,14,39,22,1:CHAR1,11,0," COORDINATE CENTRO ",1
850 CHAR1,5,2,"X=":PRINTX:CHAR1,31,2,"Y=":PRINTY
860 CHAR1,5,4,"IMMETTI NUOVA 'X' (30-289)":INPUTX$
870 CHAR1,5,6,"IMMETTI NUOVA 'Y' (30/170)":INPUTY$
880 X=VAL(X$):Y=VAL(Y$):SLEEP1:SCNCLR1:GRAPHIC1:WINDOW0,0,39,24,1:BEND:GOTO480
890 IFA$="6"THENBEGIN:SCNCLR:CHAR1,16,0,"IMMETTI DATI"
900 CHAR1,10,4,"CENTRO":CHAR1,18,3,"X 35/284":INPUTX$:IFVAL(X$)<35ORVAL(X$)>284T
HEN900
910 CHAR1,18,5,"Y 30/169":INPUTY$:IFVAL(Y$)<30ORVAL(Y$)>269THEN910
920 CHAR1,10,8,"RAGGI":CHAR1,18,7,"R 15/80":INPUTR$:IFVAL(R$)<15ORVAL(R$)>80THEN
920
930 CHAR1,18,9,"R1 0/80":INPUTR1$:IFVAL(R1$)>80THEN930
940 CHAR1,10,12,"ROTAZIONE IN. 0/180":INPUTRT$:IFVAL(RT$)>180THEN940
950 CHAR1,10,14,"PASSO ROTAZIONE 1/18":INPUTP$:IFVAL(P$)>18ORVAL(P$)<1THEN950
960 CHAR1,10,16,"DEFINIZIONE 1/180":INPUTS$:IFVAL(S$)<1ORVAL(S$)>180THEN960
970 CHAR1,10,18,"INCREMENTO X 3/-3":INPUTZ$:IFVAL(Z$)<-3ORVAL(Z$)>3THEN970
980 CHAR1,10,20,"INCREMENTO Y 3/-3":INPUTZ1$:IFVAL(Z1$)<-3ORVAL(Z1$)>3THEN970
990 X=VAL(X$):Y=VAL(Y$):RT=VAL(RT$):P=VAL(P$):S=VAL(S$):R=VAL(R$):R1=VAL(R1$):Z=
VAL(Z$):Z1=VAL(Z1$)
1000 SCNCLR1:GRAPHIC1:ZZ=0:BEND:GOTO480
1010 GOTO740
1020 GETKEYR$
```

X= 105 Y= 120 RT= 79 P= 12
R= 63 R1= 24 S= 14 Z=-3 Z1= 3

X= 196 Y= 130 RT= 41
P= 4 R= 74 R1= 32
S= 3 Z=-1 Z1=-2

```
1030 IFR$=" "THENRETURN
1040 IFR$="*"THENU$="*":RETURN
1050 REM -------ROUTINE HARDCOPY--------
1060 FAST:FORI=5392TO5740
1070 READA:POKEI,A
1080 NEXT:SLOW:RETURN
1090 DATA169,4,170,160,255,32,186,255
1100 DATA169,0,32,189,255,32,192,255
1110 DATA162,4,32,201,255,169,8,32
1120 DATA210,255,169,13,32,210,255,169
1130 DATA0,141,3,21,169,6,141,4
1140 DATA21,169,0,141,0,21,141,1
1150 DATA21,169,128,141,5,21,173,3
1160 DATA21,141,2,21,173,0,21,141
1170 DATA6,21,173,1,21,41,248,141
1180 DATA7,21,169,0,141,8,21,141
1190 DATA9,21,173,2,21,74,74,74
1200 DATA170,240,23,169,0,168,72,24
1210 DATA152,105,64,168,104,105,1,72
1220 DATA202,208,244,140,9,21,104,141
1230 DATA8,21,173,2,21,41,7,141
1240 DATA10,21,173,1,21,41,7,141
1250 DATA11,21,169,7,56,237,11,21
1260 DATA141,11,21,24,169,0,109,9
1270 DATA21,168,169,32,109,8,21,170
1280 DATA152,109,7,21,168,138,109,6
1290 DATA21,170,152,109,10,21,141,13
1300 DATA21,138,105,0,141,12,21,172
1310 DATA13,21,173,12,21,133,252,169
1320 DATA0,133,251,177,251,141,14,21
1330 DATA169,1,174,11,21,240,4,10
1340 DATA202,208,252,45,14,21,240,21
1350 DATA173,2,21,56,237,3,21,141
1360 DATA15,21,169,1,174,15,21,240
1370 DATA4,10,202,208,252,24,109,5
1380 DATA21,141,5,21,173,3,21,24
1390 DATA109,4,21,238,2,21,205,2
1400 DATA21,48,3,76,82,21,173,5
1410 DATA21,32,210,255,24,173,1,21
1420 DATA105,1,141,1,21,173,0,21
1430 DATA105,0,141,0,21,169,1,205
1440 DATA0,21,208,7,169,64,205,1
1450 DATA21,240,3,76,65,21,169,13
1460 DATA32,210,255,173,3,21,24,105
1470 DATA7,141,3,21,201,196,176,3
1480 DATA76,57,21,201,203,240,8,169
1490 DATA3,141,4,21,76,57,21,169
1500 DATA15,32,210,255,169,13,32,210
1510 DATA255,32,210,255,169,4,32,195
1520 DATA255,32,204,255,96
1530 RETURN
1540 GRAPHIC0,1:CHAR1,13,12,""
1550 PRINTERR$(ER);" IN "EL:SLEEP2
1560 RESUME610
1570 END
```

X= 124 Y= 142 RT= 49 P= 5 R= 73
R1= 4 S= 58 Z= 0 Z1=-1

X= 201 Y= 135 RT= 42 P= 4 R= 69
R1= 4 S= 1 Z=-1 Z1=-2

X= 100 Y= 100 RT= 50 P= 5 R= 80
R1= 40 S= 30 Z= 1 Z1= 1

```
100 REM C/128, C/16, PLUS 4
110 REM PROGRAMMA DI GRAFICA: RANDOM PROJECT
120 REM BY M. & S. CIURLI - ROMA
130 COLOR1,8:COLOR4,1:COLOR0,1:GOSUB590
140 SCNCLR
150 TRAP560:GRAPHIC0
160 REM ---------MENU' OPZIONI-------
170 COLOR4,1:COLOR0,1
180 SCNCLR:PRINT"■                                                  ■";
190 PRINT"■           RANDOM PROJECT                   ■";
200 PRINT"■                                            ■■";
210 PRINT"ESEM. 30  10 50 1 .1";
220 PRINT"      50 -30 75 2 .1"
230 PRINT"      28  25 20 5 .2";
240 PRINT"      50 -20 45 5 .1"
250 PRINT"      36  24 12 3 .1";
260 PRINT"      72 -48 24 2 .2"
270 PRINT"      50  12  3 4 .3";
280 PRINT"     -62 -32 12 4 .2"
290 PRINT"     -50 -12 -3 4 .3";
300 PRINT"     -58 -25  8 4 .2"
310 PRINT"      50 -48 12 4 .2";
320 PRINT"     -50 +48-12 4 .2"
330 PRINT"■■                                            ■";
340 INPUT"■ PRIMO   RAGGIO...";R0:A(0)-R0
350 INPUT"■ SECONDO RAGGIO...";R1:A(1)-R1
360 INPUT"■ TERZO   RAGGIO...";R2:A(2)-R2
370 INPUT"■ NUMERO GIRI......";NG:A(3)-NG
380 INPUT"■ PASSO............";PA:A(4)-PA
390 CHAR1,0,23," '←'-NUOVE VARIABILI. 'SPACE'-RUN",1
400 GETKEYA$:IFA$-"←"THEN150
410 X1-9999
420 REM----------ROUTINE CALCOLO------
430 SCNCLR :GRAPHIC1,1
440 FOR A-0TO2*π*NG+PA STEP PA
450 GET A$:IF A$-" "THEN GRAPHIC0,1:GOTO180
460 XC-SIN(A)*(R2+R1)+160
470 X -XC+R0*COS(A*R2/R1)
480 YC-SIN(A)*(R2+R1)+100
490 Y -YC+R0*SIN(A*R2/R1)
500 IF X1-9999 THEN 520
510 DRAW1,XK,YKTOXC,YCTOX,YTOX1,Y1
520 X1-X:Y1-Y:XK-XC:YK-YC:NEXT
530 GETKEYA$
540 IFA$-" "THENGRAPHIC0:GOTO150
550 GOTO530
560 GRAPHIC0::PRINT"■ERRORE : PREMI UN TASTO."
570 GETKEYA$:GOTO150
580 REM-------INTRODUZIONE-------------
590 V-13:PRINTCHR$(147)
600 GRAPHIC1,1
610 FORK--159TO159STEP6
620 Y-199:X-K+V*K:IFABS(X)>160-(X<1)THENX-SGN(X)*160+(X<1):Y-111+(X-K)*80/(V*K)
630 DRAW1,160-K,111TO160-X,Y
640 DRAW1,160-K,90TO160-X,199-Y
650 NEXTK
660 F-V↑(1/6):Y-F
670 Y-Y*F:IFY>93THENGETKEYA$:RETURN
680 DRAW1,0,108+YTO319,108+Y
690 DRAW1,0,94-YTO319,94-Y
700 GOTO670
710 END
```

Caccia alla mosca

Con l'approssimarsi della buona stagione, i fastidiosi insetti cercheranno in tutti i modi di arrecare disturbo...

di Paolo Scalabrini

Scopo del gioco è abbattere le mosche che infestano il vostro giardino. Avete a disposizione un potente insetticida che emette un massimo di 15 micidiali nuvolette.

Finite le "munizioni" il gioco termina. Le scariche dell'insetticida si controllano con la tastiera (tasto cursore in alto=sinistra; cursore a destra=destra; barra spaziatrice=fuoco). Ogni mosca abbattuta vale cinque punti.

Come digitarlo

Il listato adotta lo standard suggerito su C.C.C. N. 39: è possibile digitare prima il cuore del programma (primo listato) e poi i Data (secondo listato) relativi agli sprite, che sono in numero di tre.

Sottolineamo che è necessario digitare il secondo listato mentre il primo è presente nella memoria del C/64.

Il programma è "farcito" di REM che lo commentano diffusamente.

Come funziona

Per scopo didattico riportiamo la descrizione del listato che compare in queste pagine:

10-15 definisce le variabili (notare la modularità)
19-50 disegna il "campo di battaglia" e posiziona gli sprite

posizione	val.
fermo	127
su	126
giu'	125
destra	119
sinistra	123
sin+su	122
sin+giu'	121
des+su	118
des+giu'	117
fire	111
fire+su	110
fire+giu'	109
fire+sin	107
fire+des	103
fire+su+sin	106
fire+giu'+sin	105
fire+su+des	102
fire+giu'+des	101

52-90 salta alle varie subroutine
120-150 lettura tastiera
500 posiziona lo sprite
600-651 routine che controlla che gli sprite non escano dallo schermo
800-830 muove la nuvoletta spray
820 controlla la collisione tra mosca e nuvoletta
837 controlla il numero di colpi rimasti a disposizione
1000-1200 disegna gli sprite
2000-2100 muove casualmente la mosca
3000-3040 indica che il gioco è finito e ricomincia
3500-3530 visualizza punteggio e il numero di colpi rimasti a disposizione del giocatore
4000-4010 fa precipitare la mosca

I lettori che lo desiderano potranno gestire i movimenti per mezzo del joystick apportando un minimo di modifiche.

Per facilitare le stesura della subroutine relativa sarà certamente utile la tabella pubblicata che, evitando di ricorrere a complicati calcoli di AND logici e OR esclusivi, mostra i valori ottenuti leggendo la locazione 56320 in seguito ai diversi spostamenti dell'assicella del joystick in Porta 2.

SCHEDA TECNICA

Videogame a scopo fondamentalmente didattico
Anche i programmi pubblicati in queste pagine sono contenuti nel disco "Directory" di questo mese.
Idoneo per il C/64 e non adattabile ad altri computer Commodore
Consigliata la rilettura del fascicolo N.39.
Software particolarmente idoneo per gli utenti principianti

```
   0 REM CACCIA ALLA MOSCA
   1 REM GIOCO PER C/64
   2 REM DI PAOLO SCALABRINI
   3 REM CAMPOMORONE (GE)
   4 :
  10 X=160:Y=100:X1=160:Y1=220:R
     EM  VARIABILI POSIZIONE SPR
     ITES
  11 V=53248:S=54296:REM  DEFINI
     ZIONE VIDEO E SUONO
  12 Q=0:T=0:REM  VARIABILI DI C
     OMODO
  13 M=0:C=15:REM  MOSCHE COLPIT
     E - COLPI DISPONIBILI
  14 IN$="":REM  INPUT VERSIONE
     TASTIERA
  15 JK=0:REM  INPUT VERSIONE JO
     YSTICK
  19 POKE  V+21,3:POKE  V,X:POKE
       V+1,Y:PRINT "[CLEAR]":REM
       ACCENSIONE SPRITES E POS.
     MOSCA
  20 FOR  T=1984  TO  2023:POKE
      T,160:NEXT:REM  DISEGNA PA
     VIMENTO
  30 FOR  T=56256  TO  56295:POK
     E  T,5:NEXT:REM  COLORE VER
     DE
  40 POKE  53280,0:POKE  53281,1
     4:REM  BORDO NERO, SFONDO A
     ZZURRO
  50 POKE  V+2,X1:POKE  V+3,Y1:R
     EM  POSIZIONA LO SPRAY
  52 GOSUB  3500
  60 GOSUB  1000
  70 GOSUB  1100
  80 GOSUB  1200
  90 GOSUB  2000
 120 GET  IN$:REM  LETTURA DA TA
     STIERA
 130 IF  IN$=" " THEN  GOSUB  80
     0:REM  SE FIRE...
 140 IF  IN$="[DOWN]" THEN  X1=X
     1-10:GOSUB  650:REM  SE SIN
     ISTRA...
 150 IF  IN$="[RIGHT]" THEN  X1=
     X1+10:GOSUB  640:REM  SE DE
     STRA...
 500 POKE  V+2,X1:GOTO  90:REM
     POSIZIONA LO SPRAY A NUOVA
     LOCAZIONE
 599 REM  CONTROLLO CHE GLI SPRI
     TES NON ESCANO DALLO SCHERM
     O
 600 IF  Y<50  THEN  Y=50
 601 RETURN
 610 IF  Y>150  THEN  Y=150
 611 RETURN
 620 IF  X>255  THEN  X=255
 621 RETURN
 630 IF  X<24 THEN  X=24
 631 RETURN
 640 IF  X1>255  THEN  X1=255
 641 RETURN
 650 IF  X1<24  THEN  X1=24
 651 RETURN
 799 REM  MUOVO LA NUVOLETTA DI
     SPRAY
 800 POKE  V+21,7:POKE  V+4,X1:P
     OKE  V+5,Y1
 810 FOR  T=Y1  TO  0  STEP -10:
     POKE  V+5,T:GOSUB  2000
 819 REM  CONTROLLA LA COLLISION
     E
 820 IF  PEEK(V+30)=5  THEN  400
     0
 830 POKE  V+30,0:NEXT:C=C-1
 835 GOSUB  3500
 836 REM  SE NON CI SONO PIU' CO
     LPI...
 837 IF  C=0  THEN  3000
 840 RETURN
 999 END
1000 POKE  2040,0:POKE  V+39,0:R
     ETURN:REM  SPRITE 1
1100 POKE  2041,0:POKE  V+40,0:R
     ETURN:REM  SPRITE 2
1200 POKE  2042,0:POKE  V+41,0:R
     ETURN:REM  SPRITE 3
1999 REM  MOVIMENTO CASUALE DELL
     A MOSCA
2000 A=INT(RND(1)*8)+1:POKE  S,1
     5:REM  VOLUME MASSIMO
2010 IF  A=1  THEN  Y=Y+15:GOSUB
       610
2020 IF  A=2  THEN  Y=Y-15:GOSUB
       600
2030 IF  A=3  THEN  X=X+15:GOSUB
       620
```

```
2040 IF  A=4  THEN  X=X-15:GOSUB
      630
2050 IF  A=5  THEN  X=X-15:Y=Y-1
     5:GOSUB  630:GOSUB  600
2060 IF  A=6  THEN  X=X-15:Y=Y+1
     5:GOSUB  630:GOSUB  610
2070 IF  A=7  THEN  X=X+15:Y=Y-1
     5:GOSUB  620:GOSUB  600
2080 IF  A=8  THEN  X=X+15:Y=Y+1
     5:GOSUB  620:GOSUB  610
2090 POKE  V,X:POKE  V+1,Y:POKE
      S,0:REM  POSIZIONO MOSCA E
      AZZERO VOLUME
2100 RETURN
2999 REM  GAME OVER
3000 FOR  T=1  TO  2000:NEXT
3010 POKE  V+21,0:PRINT"[CLEAR]"
3020 FOR  T=1  TO  6:PRINT CHR$(
     17):NEXT
3030 PRINT  TAB(15)"[BLEU]GAME
     OVER"
3040 FOR  T=1  TO  5000:NEXT: RU
     N
3499 REM  VISUALIZZA PUNTI E COL
     PI RIMASTI
3500 PRINT "[HOME]" TAB(32)"[NER
     O]PUNTI[BIANCO]";M
3510 PRINT  TAB(32)"[NERO]COLPI[
     BIANCO]";C;:IF  C<10 THEN
     PRINT"[LEFT] "
3520 IF  C=0  THEN  3000
3530 RETURN
3999 REM  FACCIO PRECIPITARE LA
     MOSCA
4000 POKE  V+21,3:M=M+5:C=C-1:FO
     R  T=PEEK(V+1) TO  255:POKE
      V+1,T:NEXT:GOSUB  3500
4005 POKE  V+5,0
4010 GOTO  90
4020 END
```

```
 999 END
1000 POKE  2040,13:POKE  V+39,0:
     FOR  T=0  TO  62:READ  Q:PO
     KE  832+T,Q:NEXT:RETURN
1010 DATA  0,0,0,96,0,24,152,0,1
     00
1020 DATA  132,0,132,130,1,4,66,
     1,8
1030 DATA  65,2,8,33,50,16,24,20
     4,96
1040 DATA  5,206,128,3,207,0,3,1
     35,0
1050 DATA  1,50,0,0,252,0,3,51,0
1060 DATA  4,120,128,8,120,64,16
     ,0,32
1070 DATA  0,0,0,0,0,0,0,0,0
1100 POKE  2041,14:POKE  V+40,2:
     FOR  T=0  TO  62:READ  Q:PO
     KE  896+T,Q:NEXT:RETURN
1110 DATA  192,0,0,207,0,0,49,12
     8,0
1120 DATA  35,192,0,71,224,0,79,
     240,0
1130 DATA  95,152,0,127,12,0,62,
     6,0
1140 DATA  28,3,0,12,1,128,6,0,1
     92
1150 DATA  3,0,96,1,128,112,0,19
     2,248
1160 DATA  0,97,240,0,51,224,0,3
     1,192
1170 DATA  0,15,128,0,7,0,0,2,0
1200 POKE  2042,200:FOR Z=0  TO
      62:READ  Q:POKE  64*200+Z,
     Q:NEXT:POKE  V+41,1:RETURN
1210 DATA  28,0,0,54,0,0,255,0,0
1220 DATA  183,0,0,254,0,0,56,0,
     0
1230 DATA  48,0,0,0,0,0,0,0,0
1240 DATA  0,0,0,0,0,0,0,0,0,0,0
     ,0
1250 DATA  0,0,0,0,0,0,0,0,0,0,0
1260 DATA  0,0,0,0,0,0,0,0,0,0
1270 DATA  0,0,0
```

CORK

Una questione di puntatori

Alterando opportunamente alcuni puntatori che sovrintendono al drive, al Basic e al sistema operativo, si ottengono effetti sorprendenti

Files (24184/24345)

L'idea di creare una routine che visualizzi la directory di un file non è nuova, ma dato che lo scopo di queste pagine è quello di divulgare utility interamente rilocabili, e sopratutto di spiegare come funzionano, è sembrato opportuno soffermarsi sul formato di registrazione dell'utilissimo catalogo dei programmi, posto sulla traccia 18 dei dischi registrati con i drive di formato 2A (1541, 1571, 2040, 4040 e compatibili).

Riferendosi al programma Basic, fornito su disco nella confezione del drive (Test Demo), questo appare oscuro e farcito di una serie di GET#; per meglio comprenderlo bisogna sapere come è immagazzinato un programma Basic in memoria.

I puntatori che indicano la prima locazione di memoria riservata al Basic, sono quelli allocati in 43 e 44; se infatti, appena acceso il C/64, scrivete...

*PRINT PEEK(43)+PEEK(44)*256*

...otterrete 2049, che è la locazione di inizio dell'area dei programmi Basic.

Le prime due locazioni di ogni linea Basic contengono il puntatore alla linea successiva, ovvero se chiamiamo "A" la locazione di inizio della linea, con Peek(a)+Peek(a+1)*256, otteniamo il numero del byte dell'inizio della linea successiva.

Se le prime due locazioni della linea contengono entrambe zero, vuole dire che il programma è terminato; in caso contrario i due byte successivi rappresentano il numero della linea Basic.

E' poi presente una sequenza di numeri, cioè i byte che compongono la linea stessa, che termina sempre con zero.

Quando si registra un file su disco, con l'istruzione Save, verranno salvati anche due byte (posti prima dell'inizio vero e proprio del file) che rappresenteranno la prima locazione di memoria del file registrato.

La Directory viene strutturata come un qualunque programma Basic, in cui i numeri di linea rappresentano i blocchi occupati dal programma.

Quando dobbiamo caricare la directory è necessario scartare i due byte registrati all'inizio e tutti i byte di link che normalmente indicherebbero l'inizio della linea successiva; quando, nella sequenza di byte del nome, troviamo uno zero, vuol dire che la linea è terminata e i byte successivi rappresentano il numero della linea successiva. La Directory, caricata con: Load"$",8 sembra simile ad un qualsiasi programma Basic; se, però, provate a digitare...

Load"$",8,1

...vi accorgerete della differenza!

Come si usa

Per utilizzare la routine proposta è necessario digitare una delle due forme sintattiche seguenti:

SYS (XXXX)

oppure

SYS (XXXX) Y$

in cui XXXX è la locazione di memoria da cui inizia, mentre Y$, facoltativo, indica il comando. Si noti, dopo l'indirizzo della routine, l'assenza del carattere virgola (,) che, se presente, causerebbe un Syntax Error.

Per il comando sono utilizzabili entrambi i patterns (asterisco e punto interrogativo); per caricare, ad esempio, il nome di tutti i file che iniziano per "A", dovrete utilizzare come pattern "A*".

Al termine della visualizzazione è necessario premere un tasto qualsiasi per far riapparire il cursore.

```
1000 PRINTCHR$(147)"FILES"
1010 PRINT"SERVE A CARICARE LA D
     IRECTORY"
1020 PRINT"SENZA CANCELLARE I PR
     OGRAMMI"
1030 PRINT"IN MEMORIA"
1040 PRINT:PRINT"SYS XXXX
1050 PRINT"SYS XXXX"CHR$(34)"A*"
     CHR$(34)
1060 PRINT"SYS XXXX"CHR$(34)"??B
     A"CHR$(34)
1070 RETURN
1080 DATA  169,096,032,195,255,0
     32,121
1090 DATA  000,208,006,169,001,1
     33,251
1100 DATA  208,025,032,158,173,0
     32,130
1110 DATA  183,152,133,251,230,2
     51,170
1120 DATA  240,011,160,000,177,0
     34,153
1130 DATA  065,003,200,202,208,2
     47,169
1140 DATA  036,141,064,003,169,0
     96,162
1150 DATA  008,168,032,186,255,1
     65,251
1160 DATA  162,064,160,003,032,1
     89,255
1170 DATA  032,192,255,032,183,2
     55,041
1180 DATA  128,240,005,162,005,1
     08,000
1190 DATA  003,162,096,032,198,2
     55,160
1200 DATA  003,132,251,032,207,2
     55,133
1210 DATA  252,164,144,208,047,0
     32,207
1220 DATA  255,164,144,208,040,1
     64,251
1230 DATA  136,208,233,166,252,0
     32,205
1240 DATA  189,169,032,032,210,2
     55,032
1250 DATA  207,255,166,144,208,0
     18,170
1260 DATA  240,006,032,210,255,0
     24,144
1270 DATA  240,169,013,032,210,2
     55,160
1280 DATA  002,208,198,169,096,0
     32,195
1290 DATA  255,032,204,255,165,1
     97,201
1300 DATA  064,240,250,169,000,1
     33,198
1310 DATA  096,-1,22771
```

```
 ORIGIN:$C000
       LDA #$60     chiude il file #96
       JSR $FFC3    se e' aperto.
       JSR $0079    se non c'e' nessun
       BNE *C010    parametro,
       LDA #$01     la lunghezza del nome
       STA $FB      sara' 1,
       BNE *C029
*C010  JSR $AD9E    altrimenti prende la
       JSR $B782    stringa,
       TYA
       STA $FB      calcola la lunghezza
       INC $FB
       TAX
       BEQ *C029
       LDY #$00      e la mette nell'area
*C020  LDA ($22),Y di memoria temporanea
       STA $0341,Y
       INY
       DEX
       BNE *C020
*C029  LDA #$24     aggiunge $ prima del
       STA $0340    nome del file,
       LDA #$60     il canale 96,con
       LDX #$08     periferica disco e
       TAY
       JSR $FFBA
       LDA $FB
       LDX #$40     nome contenuto
       LDY #$03     nell'area temporanea,
       JSR $FFBD
       JSR $FFC0    viene aperto
       JSR $FFB7    se al controllo
       AND #$80     manca la periferica
       BEQ *C04E
       LDX #$05     device not present
       JMP ($0300) error.
*C04E  LDX #$60     dispone il canale
       JSR $FFC6    per input
       LDY #$03     apre un loop di 3 byte
*C055  STY $FB
       JSR $FFCF    prende un carattere
       STA $FC      e lo mette in memoria
       LDY $90      controlla la
       BNE *C08F    condizione d'errore
       JSR $FFCF    prende un altro byte
       LDY $90      e controlla ancora
       BNE *C08F    lo status
       LDY $FB      se il loop e' aperto,
       DEY
       BNE *C055    lo chiude
       LDX $FC      ora contiene il numero
       JSR $BDCD    dei blocchi in AX e
                    lo stampa
       LDA #$20     stampa di seguito uno
       JSR $FFD2    spazio
*C076  JSR $FFCF    prende un byte
       LDX $90      e controlla lo status
       BNE *C08F
       TAX          se il carattere e' 0,
```

```
         BEQ *C086   prossimo filename
         JSR $FFD2   altrimenti lo stampa
         CLC         passa al prossimo
         BCC *C076   byte
*C086    LDA #$0D    stampa un ritorno
         JSR $FFD2   carrello
         LDY #$02    e dispone un loop
         BNE *C055   di due bytes
*C08F    LDA #$60    chiude il file
         JSR $FFC3
         JSR $FFCC
*C097    LDA $C5     e aspetta la pressione
         CMP #$40    di un tasto.
         BEQ *C097
         LDA #$00    azzera il buffer di
         STA $C6     tastiera e' torna
         RTS         al basic
```

Delete (versione 2) (24346/24492)

Spesso, durante la stesura di un programma Basic, capita di dover cancellare un gruppo di linee, e se l'elenco è lungo, lo spreco di tempo è enorme; è utile, allora, disporre di una routine che esegua automaticamente, e velocemente, il lavoro.

L'utility pubblicata, piuttosto breve, risulta più versatile di quella, analoga, pubblicata sul N.41. Su come funziona non c'è molto da dire: la routine calcola le locazioni relative alla prima e alla linea successiva all'ultima riga Basic, spostando indietro la parte restante del programma e riaggiornando i puntatori.

Si è preferito sfruttare la routine contenuta nel sistema operativo, posta a partire dalla locazione $A613, che si occupa della ricerca di una determinata linea in un programma; tale routine è utilizzata, per esempio, per determinare l'argomento di Goto.

Per il suo funzionamento bisogna allocare il numero di linea in $14-$15 e saltare alla routine stessa; se il carry è settato vuol dire che la linea è presente nel programma e la locazione di partenza è contenuta in $5F-$60; se il carry, al contrario, è posto a zero, la linea non è presente, e in $5F-$60 sarà contenuto il byte di inizio di quella con numerazione immediatamente più alta.

Come si usa

Le sintassi disponibili sono quattro:

SYS(XXXX)A
SYS(XXXX)A,B
SYS(XXXX)A,
SYS(XXXX),B

in cui "A" e "B" sono il numero della prima e dell'ultima linea.

Molta attenzione bisogna prestare al fatto che l'indirizzo va posto fra parentesi e che, dopo la chiusura, non deve esser presente la virgola, pena la cancellazione accidentale di gruppi di linee Basic.

La prima sintassi cancella la linea A, la seconda le linee comprese tra A e B (estremi inclusi), la terza "A" e le linee successive, mentre l'ultima tutte le linee fino a "B".

```
1000  PRINTCHR$(147)"DELETE"
1010  PRINT"SERVE PER CANCELLARE
      LINEE"
1020  PRINT"IN UN PROGRAMMA."
1030  PRINT:PRINT"SYS (XXXX)A"
1040  PRINT"SYS (XXXX)A,B"
1050  PRINT"SYS (XXXX)A,"
1060  PRINT"SYS (XXXX),B"
1070  RETURN
1080  DATA  032,121,000,208,003,0
      76,008
1090  DATA  175,144,004,201,044,2
      08,037
1100  DATA  032,107,169,032,019,1
      66,032
1110  DATA  121,000,240,012,201,0
      44,208
1120  DATA  022,032,115,000,032,1
      07,169
1130  DATA  208,014,165,020,005,0
      21,208
1140  DATA  011,169,255,133,020,1
      33,021
1150  DATA  208,003,076,008,175,1
      65,095
1160  DATA  166,096,133,036,134,0
      37,032
1170  DATA  019,166,144,014,160,0
      01,177
1180  DATA  095,240,008,170,136,1
      77,095
1190  DATA  133,095,134,096,165,0
      36,056
1200  DATA  229,095,170,165,037,2
      29,096
1210  DATA  168,176,030,138,024,1
      01,045
1220  DATA  133,045,152,101,046,1
      33,046
1230  DATA  160,000,177,095,145,0
      36,200
1240  DATA  208,249,230,096,230,0
      37,165
1250  DATA  046,197,037,176,239,0
      32,051
1260  DATA  165,165,034,166,035,0
      24,105
1270  DATA  002,133,045,144,001,2
      32,134
1280  DATA  046,032,089,166,076,1
      16,164
1290  DATA  -1,15424
```

```
ORIGIN:$C000
       JSR $0079   se non ci sono
       BNE *C008   parametri
       JMP $AF08   syntax error
*C008  BCC *C00E   se non c'e un numero
       CMP #$2C    ci deve essere
       BNE *C033   una virgola
*C00E  JSR $A96B   prende un numero
       JSR $A613   e calcola i puntatori
                   alla linea iniziale
       JSR $0079   e se ci sono ancora
       BEQ *C025   parametri,ci deve
       CMP #$2C    essere una virgola
       BNE *C033   altrimenti errore
       JSR $0073   prende un numero
       JSR $A96B   e se ci sono ancora
       BNE *C033   parametri errore
*C025  LDA $14     se il secondo
       ORA $15     parametro e' nullo
       BNE *C036
       LDA #$FF    mette 65535 come
       STA $14     default
       STA $15     poi salta la
       BNE *C036   prossima istruzione
*C033  JMP $AF08   stampa syntax error
*C036  LDA $5F     salva i puntatori
       LDX $60     alla linea iniziale,
       STA $24     cerca la linea
       STX $25     finale e mette
       JSR $A613   i puntatori in $5f-$60
       BCC *C051   se la trova
       LDY #$01
       LDA ($5F),Y prende i puntatori
       BEQ *C051    alla linea successiva
       TAX
       DEY
       LDA ($5F),Y
       STA $5F
       STX $60
*C051  LDA $24     sottrae l'indirizzo
       SEC         della successiva
       SBC $5F     all'attuale
       TAX
       LDA $25
       SBC $60
       TAY         se il programma e'
       BCS *C07C   terminato, esce
       TXA
       CLC         addiziona il numero
       ADC $2D     di byte della linea
       STA $2D     complementato a due
       TYA         (quindi sottrae)
       ADC $2E     ai puntatori di fine
       STA $2E     programma
       LDY #$00
*C06B  LDA ($5F),Y spostando la memoria,
       STA ($24),Y cancella le linee
       INY
       BNE *C06B
       INC $60
       INC $25
       LDA $2E
       CMP $25
       BCS *C06B
*C07C  JSR $A533   esegue un relink al
       LDA $22     programma basic
       LDX $23
       CLC
       ADC #$02    setta la fine
       STA $2D     del programma basic.
       BCC *C08B
       INX
*C08B  STX $2E
       JSR $A659   esegue clr
       JMP $A474   torna al basic.
```

Reset lock (24493/24572)

Ogni volta che accendiamo il computer, o che premiamo il tasto di reset, viene eseguita una routine chiamata routine di reset; quando premiamo il tasto Restore viene eseguita una routine chiamata NMI (interrupt non mascherabile) che controlla se è premuto contemporaneamente anche il tasto Run/Stop e, in caso affermativo, attiva la routine chiamata BRK.

Nel sistema operativo del Commodore 64 è anche presente un controllo che permette di determinare se è inserito un cartridge nell'apposita porta, in modo da evitare che si possa interrompere il funzionamento della stessa, e tornare al Basic, e anche per eseguire l'autostart all'accensione del calcolatore.

Il computer si "accorge" se è presente una cartuccia verificando se, a partire da $8004, è presente la stringa CBM80, in cui, al valore Ascii dei primi tre caratteri, è sommato 128.

La locazione di inizio della nuova routine di Restore verrà messa a $8000-$8001, mentre quella di NMI a $8002-$8003.

Possiamo quindi manipolare la memoria facendo credere al computer che sia presente un cartridge e gestire, di conseguenza, nostre routine di NMI e Reset; sarà possibile, ad esempio, cambiare la routine di Reset in modo che aggiorni i puntatori ma non cancelli il programma presente in memoria evitando così che, in caso di malfunzionamento, lo stesso venga cancellato in seguito alla pressione di reset; potremmo anche programmare il tasto di Restore in modo che fermi l'esecuzione del programma alla prima pressione e che lo faccia ripartire alla seconda, ma vi sono tantissime applicazioni che lasciamo alla vostra fantasia.

Come si usa

Sono due le sintassi usate per questa routine:

SYS XXXX
SYS XXXX,RES,NMI

in cui XXXX è la locazione d'inizio del programma. La prima sintassi permette di restaurare la situazione

standard di Reset e Restore; la seconda serve a porre la stringa di controllo in memoria e a determinare l'inizio delle nuove routine di Reset (RES) e Restore (NMI).

Se vogliamo lasciare invariata una delle due routine, cambiando l'altra, dovremo assegnare il rispettivo valore standard:
64751: nel caso della routine di reset
65118: per quella di NMI.

Come applicazione pratica potete indirizzare la routine Files, pubblicata in queste stesse pagine, alla quale è però necessario modificare le ultime due righe di Data come segue:

1290 data 255,032,204,255,032,163,253
1300 data 076,123,227
1310 data -1,21932

La modifica consiste nell'eliminare la routine di attesa pressione tasto (vedi disassemblato relativo) e nell'aggiungere due "salti":

JSR $FDA3
JMP $E37B

Tale modifica è necessaria perchè il computer ha bisogno di sistemare alcune cose se viene attivata la routine di Restore e di Reset.

In pratica, digitando...

SYS 24493,24184,24184

...tutte le volte che premerete il tasto Restore verrà visualizzata la Directory del disco presente nel drive.

Il disassemblato

Il disassemblato commentato relativo alle tre routine (ideate e scritte da Fabio Sorgato), sarà sicuramente apprezzato da coloro che studiano il Linguaggio Macchina. Da notare, soltanto, che le tre routine sembrano allocate a partire da $C000; naturalmente, grazie alla loro rilocabilità, è possibile trascriverle in una qualsiasi zona Ram del C/64.

```
1000 PRINTCHR$(147)"RESET LOCK"
1010 PRINT"SERVE A MODIFICARE LE
     ROUTINE"
1020 PRINT"DI RESET E NMI"
1030 PRINT:PRINT"SYS XXXX
1040 PRINT"SYS XXXX,RES,NMI"
1050 PRINT"STANDARD RES:64751"
1060 PRINT"STANDARD NMI:65118"
1070 RETURN
1080 DATA  032,121,000,208,011,1
     62,008
1090 DATA  169,000,157,000,128,2
     02,016
1100 DATA  250,096,032,253,174,0
     32,138
1110 DATA  173,032,247,183,165,0
     20,141
1120 DATA  000,128,165,021,141,0
     01,128
1130 DATA  032,253,174,032,138,1
     73,032
1140 DATA  247,183,165,020,141,0
     02,128
1150 DATA  165,021,141,003,128,1
     69,195
1160 DATA  141,004,128,169,194,1
     41,005
1170 DATA  128,169,205,141,006,1
     28,169
1180 DATA  056,141,007,128,169,0
     48,141
1190 DATA  008,128,096,-1,8926
```

```
ORIGIN:$C000

      JSR $0079    se non c'e' nessun
      BNE *C010    parametro
      LDX #$08
      LDA #$00
*C009 STA $8000,X toglie i byte
      DEX          che indicano il blocco
      BPL *C009    e ritorna al basic
      RTS
*C010 JSR $AEFD    prende una virgola
      JSR $AD8A    e un parametro
      JSR $B7F7    a 16 bit che indica
      LDA $14      la nuova routine
      STA $8000    di reset e lo pone
      LDA $15      a $8001-$8001
      STA $8001
      JSR $AEFD    prende una virgola
      JSR $AD8A    e il secondo parametro
      JSR $B7F7    che indica la nuova
      LDA $14      routine di restore
      STA $8002    e che verra' messo
      LDA $15      a $8002-$8003
      STA $8003    sistema la stringa
      LDA #$C3     di controllo  CBM80
      STA $8004
      LDA #$C2
      STA $8005
      LDA #$CD
      STA $8006
      LDA #$38
      STA $8007
      LDA #$30
      STA $8008
      RTS          e ritorna al basic
```

EPIX 3001 la grande avventura continua...

A fine maggio, troverete nelle edicole il secondo numero di EPIX 3001, con cinque adventures grafiche al posto di tre: tre nuove e due vecchie. Infatti nel numero uno, per problemi di duplicazione, alcuni quadri di Jack Byteson e Cave Quest risultavano illeggibili, motivo per cui è stato deciso di fornire ai nostri utenti delle copie perfettamente funzionanti.

Per coloro che non avessero comprato il primo numero, ne riassumeremo il contenuto: trame, personaggi e storie di sapore americano, ma testi e creatività tutta italiana. Storie che offrono le emozioni proprie del romanzo e del film, ma richiedono al giocatore fantasia, abilità, memoria e pazienza per essere risolte.

Il secondo numero contiene, cinque adventures grafiche. Analizziamole:

1/ La seconda parte della spericolata avventura di Jack Byteson (un tipo alla Indiana Jones), impegnato alla ricerca di un magico amuleto che lo aiuterà a trovare la nave vichinga di Erik il Rosso. Dopo essere stato, nel primo episodio, a Chinatown eccolo nel villaggio di Bjorg in Norvegia ... le gatte da pelare sono ben tre. Al giocatore scoprirle!

2/ La prima avventura della serie "Horror", un genere nuovo, le cui caratteristiche sono racchiuse nel nome e non ha bisogno di molte spiegazioni.

Intermezzo di mezzanotte" è il nome di questo gioco notturno, ambientato in un cimitero dal quale è pressoche impossibile uscire.

Un' atmosfera cupa, vampiri che riposano in bare riposte in cripte inviolabili, conducono il giocatore in un mondo sovrannaturale, dal quale si può uscire, ma per farlo si ha bisogno, disperatamente bisogno, di qualcuno che dovrete cercare in questo lubre luogo.

Un' avventura ai confini del possibile!

3/ "Intruder", anche questa è una nuova serie ambientata in un futuro prossimo.

Gli "Intruder" sono degli androidi del tutto simili all'uomo che dall' uomo sono stati creati, ma ad esso si sono ribellati ed ora vogliono distruggere il loro creatore.

Una sottile ed inattesa vena ironica attraversail gioco rendendolo piacevolmente rilassante. Non ci sono eroi senza macchia ne peccato, ma protagonisti caricature di loro stessi, nelle piccole come nelle grandi cose, nevrotici come gli uomini d' oggi.
Una vera novità nell'avventura!

4/ Riproposizione del primo episodio di Jack Byteson alla ricerca della nave vichinga.

5/ Riproposizione di: Cave Quest "Le ragazze farfalla".

EPIX 3001, per te, che ami l'avventura!

L'altra posta

Lettere, cartoline, raccomandate, pacchetti; mancano solo i telegrammi tra la corrispondenza che giunge ogni giorno in Redazione

di Alessandro de Simone

Da qualche tempo, incoraggiando i lettori con un maggior spazio dedicato alla rubrica "Domande Risposte", la mole di corrispondenza da sbrigare è aumentata considerevolmente.

Un primo provvedimento è stato preso dedicando un maggior numero di pagine alla voce dei lettori; un'altra idea è quella di dedicare altre "colonne" per affrontare argomenti sollecitati dagli stessi lettori.

Il continuare, o meno, per questa strada, verrà, come al solito, deciso dai lettori stessi: scriveteci per dire come la pensate e, soprattutto, per segnalare le vostre richieste (o insoddisfazioni).

Commodore Computer Club, non ci stancheremo di ripeterlo, è una rivista che pubblica ***esclusivamente*** *gli argomenti sollecitati dagli utenti dei computer Commodore. Chi ci ha scritto, sollecitato, pregato e... minacciato, non può negare che, in un modo o in un altro, è stato* ***sempre*** *accontentato.*

I Club

Molto successo ha incontrato il nostro suggerimento di fondare Club di appassionati di informatica. Alcuni di questi hanno fatto le cose in grande e vogliamo parlarvi di due di questi che sembrano molto attivi.

"Computer News" è un club di Zanica (Bg) che si da parecchio da fare. Pubblica, addirittura, un bollettino - giornale (distribuito soltanto ai soci iscritti) ricco di informazioni, programmi, progettini hardware e vendita per corrispondenza di hardware e software. La pubblicazione (per ora bimestrale) è ben curata e realizzata esclusivamente con output generati da Newsroom, Geos, Printshop e altre utility "creative" che dimostrano la loro validità nella stesura di pubblicazioni del genere.

L'iscrizione costa 30.000 lire: si ha diritto a sei numeri del bollettino, alla pubblicazione gratuita di annunci economici, alla tessera con la password per la banca dati (!), a sconti del 40% sul software e del 20% sull'hardware proposto sulla stessa pubblicazione.

Per informazioni rivolgersi a:

Computer News Club
Piazza della Repubblica, 16
24050 Zanica (Bg)
Tel. 035/67.22.13 (Q)
035/67.27.08

"Club Commodore Computer" è un club di Finale Ligure la cui sede è stata messa gentilmente a disposizione dal Comune della stessa cittadina. Anche questo Club ricorre al Geos e ad altri programmi consimili per la stesura del proprio bollettino. Tra le finalità sono da evidenziare quelle didattiche (corsi di alfabetizzazione informatica) e quella della diffusione di informazioni inerenti il software e l'hardware Commodore.

E' possibile incontrarsi il martedì e il mercoledì (ore 21-23) nella sede che risulta dotata di un sistema C/64 completo. La quota di iscrizione è fissata in L.50.000 annuali. Per informazioni:

Club Commodore Computer
P.zza Santa Caterina
17024 Finale Ligure (Sv)

Il giardino elettronico

Ricordate l'indovinello proposto tempo fa (C.C.C. N.39) in cui si chiedevano i parametri delle figure "D" e "L"?

Bene, per evitare che la soluzione cadesse in mani sbagliate, l'avevo trascritta su un foglio che ho poi nascosto così bene che non sono più riuscito a ritrovarlo per parecchio tempo (ecco giustificato il motivo del ritardo con cui è stato ripreso l'argomento).

La soluzione è la seguente:

Figura D:
nodi= 6, angolo= 75, tronchi= 1

Figura L:
nodi= 3, angolo= 120, tronchi= 2, "P"= 3, "N1"= 2, "N2"= 1

Il numero dei lettori che ha risposto all'indovinello è stato modesto perchè, subdolamente, nell'articolo citato avevamo scambiato di posto alcune figure...

Nella classifica di bravura e perspicacia (senza togliere nulla al merito degli... esclusi) figurano i lettori:

Matteo Salvadori di Desio (Mi) che, ignaro della "truffa" perpetrata ai danni dei lettori, ha inviato una dotta spiegazione sul perchè e sul percome i parametri non potevano essere quelli pubblicati. Mi sembri, caro Matteo, piuttosto in gamba: hai mai pensato di collaborare con la nostra rivista? Desio è (quasi) a un tiro di schioppo dalla nostra Redazione che si trova in corrispondenza della fermata "Romolo" della Metropolitana. Vieni a trovarci, se non altro per ricevere il premio che hai dimenticato di indicare nella lettera.

Anche Gaia Ottaviano, di Roma, ha fatto notare la mancata corrispondenza tra le varie figure pubblicate e, inoltre, allega una versione del programma modificato. Anche la simpatica Gaia ha dimenticato di indicare il premio desiderato, segno questo che i nostri lettori lavorano per la pura gloria (o no?). Se vuoi qualche nostro prodotto, comunque, non hai che da indicarcelo.

Fabio Brugnera di Fontanelle (Tv) sottolinea che lo scambio di figure gli ha procurato l'ulcera ma, tuttavia, si accontenta di un libro per mettere le cose a tacere. Passando ad altro, perchè invidiare Toma per la sua bravura? Io sono convinto che chiunque, impegnadosi, può raggiungere livelli elevati: ad essere veri geni, infatti, non siamo in pochi.

Paolo Valvo, di Pachino (Sr), è un quindicenne che, un po' per volta, sta diventando un vero esperto. Ma perchè non invii alcune routine per l'Enciclopedia Basic? Visto che possiedi un C/128 puoi divertirti a proporne qualcuna in modo 128. Pensaci, anzi: ti ordino di inviarmi (su disco e corredate di articolo in Easy Script) almeno tre routine; sono sicuro che saranno apprezzate dai lettori proprio come tu, ora, fai con quelle che vedi ogni mese.

Claudio Cimmino, di Nettuno (Roma), ha inviato la risposta più breve di tutti (su una cartolina postale) e lo premiamo per la sua... originalità.

Andrea Macis, di Muravera (Ca), merita in modo particolare di essere premiato perchè, avendo un C/16, ha adattato il listato al suo computer, indovinando la soluzione.

Fausto M. assicura di essere in possesso della soluzione ma che, nonostante fosse giunto di persona nei pressi della nostra Redazione, ha avuto "paura" di fare un brutta figura ed è andato via. Caro Fausto, se sapessi come mettermi in contatto con te lo farei subito: diamine! non siamo mica dei mostri, e non abbiamo intenzione di spaventare chicchessia (cappitto mi hai).

Naturalmente ringraziamo tutti gli altri che, ne sono sicuro, si rifaranno vivi con maggiore fortuna (e impegno...) in una prossima sfida.

L'ascensore

Anche questa "sfida" è stata raccolta da pochi (ma buoni). Il primo premio in assoluto se lo merita sicuramente Maurizio Migliaccio di Ambivere (Bg) che inizia la lettera chiamandomi "Fantastico de Simone" (per carità: sono un semplice iperdirettore verticale). Il bello è che non invia un programma ma un diagramma di flusso che occupa ben cinque fogli di formato A3!

Purtroppo, per motivi di spazio, non posso pubblicare, nemmeno in parte, il notevole lavoro. Non rimane che rivolgere anche a te l'invito a collaborare fattivamente con la rivista (Ambivere è a un tiro di bazooka da Milano) e a ritirare il premio che, come i veri puri di spirito, hai dimenticato di indicare...

Non male nemmeno il programma di Cristian Ghezzi di Lissone (Mi) che ha inviato un dischetto contenente programma, lettera, istruzioni e altre amenità (tranne, ovviamente, il premio desiderato in caso di "vittoria"). Poichè Lissone è a un tiro di fionda da Milano (quella grande), perchè non vieni a trovarci, soprattutto ora che hai un sistema C/64 completo?

Ringraziamo gli altri lettori (tra cui Lorenzo Morselli, di Ravenna; Massimo Arrigoni, di Milano; Eric Bollati, di Reaglie) che hanno voluto dare il proprio contributo.

A tutti gli altri lettori un consolatorio: "grazie di cuore!"

Lei non sa chi sono io

Con mia grande (e piacevole) sorpresa ho letto numerose lettere di coloro che hanno scritto esclusivamente per esprimere la propria solidarietà in merito alla poco civile missiva di un ex-utente della nostra rivista (la posta, N.39 pag.6). Tra i simpatici lettori cito Ivan Pintori e Giuseppe Gasperini, di Roma; Michele Preziosi, di Cecina; Giovanni Vinci, di Catania; Andrea Mei, di Bologna; Marco Brugnoli, di Campagnola (R.E.); Rugo Moreno di Venezia.

Tra questi, oltre ad incoraggiarci a seguire la strada intrapresa, e trascurare le eventuali critiche incivili che dovessero pervenire, alcuni ne approfittano per chiedere chi siamo e, in particolare, chi è Alessandro de Simone (cioè il sottoscritto).

Ho spesso proposto al mio editore di inserire un poster gigante che mi raffigurasse in tutto il mio splendore, ma mi è stato fatto notare che tra gli scopi di una casa editrice c'è quello di aumentare le vendite e non di contrarle.

Mi limiterò a dire, nel modesto spazio concessomi, che sono nato a Molfetta (Ba) il 19 novembre (scorpione puro) del 1947; mi trovo, quindi, della doverosa necessità di affermare che l'importante è essere giovani "dentro": per l'esterno ciò che è fatto è fatto.

Ho seguito gli studi classici (nei cinque anni regolamentari) e in quarto ginnasio sono addirittura stato rimandato in latino e greco (ma l'insegnante, naturalmente, ce l'aveva con me).

Il corso di laurea in ingegneria è stato più sofferto perchè dedicavo tempo ad altre cose. Decisi scegliere il corso di ingegneria civile perchè l'elettronica potevo studiarla per conto mio (realizzando apparecchi da autodidatta e trascurando gli altri studi) mentre sarebbe stato più difficoltoso, da autodidatta, costruire strutture in cemento armato.

Sono abbastanza sposato e ho un figlio dodicenne, Fabrizio, che, in accordo con la tradizione (secondo la quale gli interessi dei figli sono diversi da quelli dei padri), se ne infischia totalmente di computer, tastiere, Ram e Chip, procurando, di conseguenza, l'altrettanto tradizionale spina nel cuore del sottoscritto.

Per altre informazioni (ma quali, del resto?) chiedete pure.

Genitori e computer

Alcuni lettori (tra cui Massimo T. di Massa Carrara) lamentano di non potere usare liberamente il proprio computer perchè i genitori, adducendo futili motivi, tendono a limitarne l'uso; chiedono, quindi, un mio intervento in loro (dei figli, non dei genitori) favore.

Se il computer è usato pricipalmente per videogame, sono dalla parte dei genitori; un qualsiasi bel gioco dura poco, e a questa regola non sfugge nemmeno il più sofisticato dei calcolatori.

Se, però, il computer è usato per *creare* giochi, o software in generale, io ritengo che i genitori commettano un imperdonabile errore se ne limitano l'accesso. Programmare, e imparare a farlo, è una delle più costruttive attività che un giovane della nostra era abbia a propria disposizione. Per programmare è infatti indispensabile ricorrere alla razionalità e, contemporaneamente, alla fantasia; in altre parole è necessario *studiare*, in un modo o in un altro: impedire un'attività di questo tipo rappresenta, a mio parere, un omicidio culturale.

Se, poi, l'alternativa per trascorrere il tempo libero è rappresentata dal girovagare senza meta per la città, non so quale beneficio possa apportare la proibizione dello studio del Basic.

Il computer, in conclusione, se adoperato opportunamente, deve esser considerato un amico fedele e, in ogni caso, uno strumento che, volenti o nolenti, i nostri figli troveranno sul posto di lavoro (ammesso che riescano a trovarne uno...).

C'è scuola e squola

Michele Preziosi, di Cecina (Li), chiede lumi sulla possibilità di trovare lavoro nel campo del computer e suggerimenti sui corsi di studio che offrano, in concreto, maggiori possi-

bilità di inserimento nel mondo del lavoro.

Il problema è delicatissimo e, in tempi di "preiscrizioni", di notevole attualità.

Tutti, e noi per primi, scommettono sul roseo futuro dell'informatica, nella sua accezione più vasta: hardware, software, elettronica, telematica, reti, eccetera eccetera.

Sembrerebbe, quindi, che l'ideale sia di iscriversi ad una scuola per programmatori. In realtà si commetterebbe un errore gravissimo limitandosi a "seguire" la corrente. Io ritengo che l'importante sia cercare un lavoro che, principalmente, soddisfi chi lo svolga; conosco molte persone che, per ripiego, sono costrette ad accontentarsi di un posto che, pur se ben remunerato, crea un perenne senso di frustrazione ed insoddisfazione che si manifesta anche al di fuori del tempo dedicato al lavoro stesso.

Hanno un bel dire i nostri capoccioni del Palazzo che in una società moderna, in caso di necessità, bisogna attuare la mobilità del lavoro; in altre parole ci vogliono far credere che se il lavoro in miniera non rende più, ma c'è richiesta di palombari, i minatori possono benissimo cambiar mestiere e indossare gli scafandri.

Naturalmente lo stesso consiglio vale anche per gli odierni appassionati di computer: non pensate che lavorare con un calcolatore consista solo nel modificare i Link del Ready o nell'impostare un nuovo percorso per l'interrupt. Molto spesso capita di lavorare su progetti noiosissimi, poco creativi e, di conseguenza, alienanti e simili a tanti altri che, paradossalmente, avete scartato.

Incominciamo subito, quindi, col dire che il diploma di scuola media, da solo, non serve a nulla, e spero che il giovanissimo Francesco Ambrogio, di Modena, si convinca a continuare gli studi incautamente interrotti. Con Francesco, infatti, ho avuto una discussione forse troppo polemica e categorica, ma non me la sentivo assolutamente di avallare la sua decisione di limitarsi alla licenza media. Le note vicissitudini della scuola italiana, d'altra parte, hanno portato la didattica nazionale ai confini della realtà.

Io ritengo che la scuola media, come è attualmente, sia stata inventata dai Russi per renderci tutti stupidi e favorire l'invasione.

Se ci limitiamo, infatti, a guardare i programmi della Scuola Media e le intenzioni che li hanno generati, sembra di raggiungere l'apice supremo della Cultura; nella pratica, invece, una paurosa massa di tredicenni giunge alle scuole superiori con una devastante ignoranza che supera qualsiasi immaginazione. Tra le lacune che si trascinano dietro brillano la mancanza di autonomia, la pressochè totale assenza di spirito critico ed una carenza quasi incolmabile di responsabile razionalità.

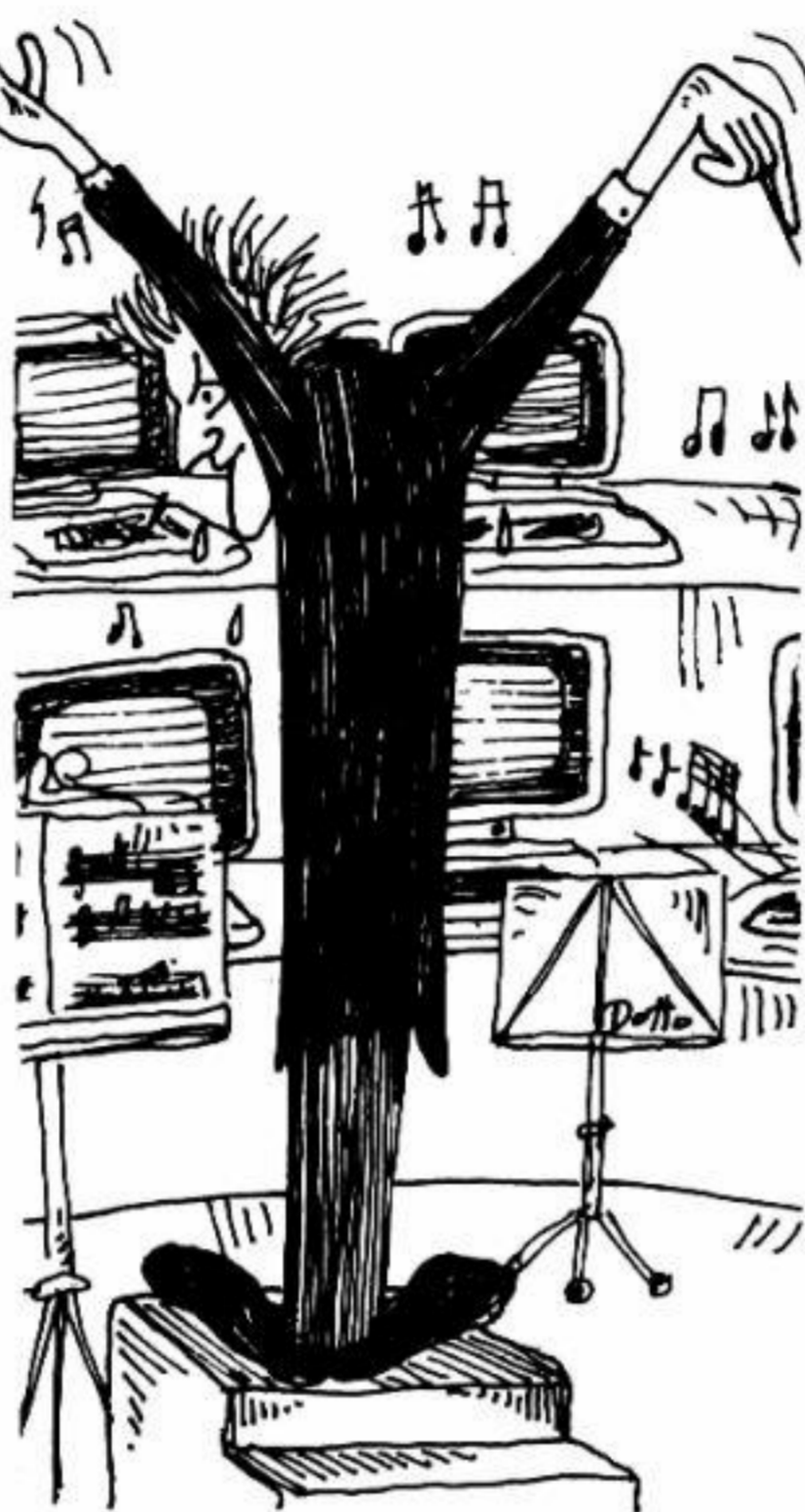
La situazione nelle scuole superiori, pur se migliore, lascia molto a desiderare, tanto che molte associazioni di industriali e commercianti (che rappresentano il mondo del lavoro) lanciano spesso pietosi appelli ai responsabili della Pubblica Istruzione affinchè, eufemisticamente, si rintracci un migliore "aggancio" con le reali esigenze produttive del Paese.

Uno dei mali della scuola è, a mio parre, la demotivazione degli insegnanti che, in una percentuale che sarebbe interessante stabilire, insegnano "per ripiego" o per tranquillità. L'aggiornamento è affidato al loro buon cuore, il mondo del lavoro è visto come un miraggio lontano sul quale ha la meglio il rispetto del Programma Ministeriale (qualunque sia la sua vetustà). Ho visto programmi di elettronica che si fermano alle val-

vole termoioniche e programmi di meccanica che nemmeno accennano alle moderne macchine a controllo numerico; e i robot? non si studia la fantascienza. Ho conosciuto insegnanti di disegno meccanico che non sanno nemmeno che cosa sia il Cad/Cam e che si rifiutano di parlarne ai propri studenti perchè "non c'è scritto nel programma". Ho sentito parlare, viceversa, di un preside che è stato duramente criticato per aver licenziato insegnanti che si dichiaravano incapaci a svolgere il pur arretrato programma di elettronica.

Nonostante il quadro non certo confortante (e forse molto personale; ma gradirei il parere dei lettori anche su questo) io ritengo che la scuola superiore italiana riesca ancora a dare moltissimo al giovane che intenda frequentarla seriamente. Per fortuna, e in ciò sono confortato dal consenso di molti giovani con i quali sono a contatto, in ogni classe della scuola superiore italiana esiste almeno un paio di insegnanti (soprattutto di italiano, di storia, di scienze; e mi perdonino gli altri che possono, comunque, protestare vibratamente) per i quali sarà valsa la pena frequentare la scuola.

Ciò che un professore (realmente motivato e appassionato per la sua materia, qualunque essa sia) riesce a trasmettere ai propri allievi, è un bene prezioso che non può essere descritto, ma solo provato. A patto, ovviamente, di essere in grado di riceverlo.

Per ciò che riguarda il corso di studi, come è facile intuire, non oso dare consigli. Mi limiterò a ricordare che il futuro offrirà calcolatori sempre più complessi, più intelligenti, dotati di linguaggi sempre più evoluti e di semplice uso. Buttandosi a capofitto in un corso di programmazione "pura" si corre lo stesso rischio corso dagli autisti di autovetture dell'inizio del secolo: all'inizio, infatti, saper guidare un'auto era piuttosto difficile e, quantomeno, inusuale. Quando, però, le auto furono di facile uso, tutti furono in grado di guidarle e l'autista, risultato non indispensabile, fu riciclato nelle fabbriche, nelle officine di riparazioni, nella guida degli

autobus, eccetera.

Leggendo gli annunci economici per ciò che riguarda la ricerca del personale, è facile notare (a parte la "fame" di venditori per attività, comunque, legate al commercio) la richiesta di programmatori esperti in campi specifici, soprattutto gestione dati e contabilità; si avverte, anche, la richiesta di esperti in telematica e in trasmissione dati in generale.

Se, quindi, avete l'opportunità di seguire un corso di informatica *applicativo* (ragionieri programmatori e simili) e se il campo è di vostro gradimento, preferitelo ad un corso per soli programmatori (a meno che, ovviamente, non abbiate altra passione che la programmazione pura).

Un ultimo avvertimento: per le scuole private (vale a dire a pagamento) consultate studenti che abbiano già seguito i corsi che anche voi intendete seguire: vi sono, infatti, scuole private di notevole serietà, altri casi, purtroppo, nascondono mostruosi bidoni. Intervistare i diplomati *prima* di iscriversi ad una scuola (privata o pubblica), risulta, ad ogni buon conto, una preziosa fonte di informazioni...

Abbasso i listati

Calogero Bonosio, di Enna, ritiene che la nostra pubblicazione rischia di stancare molti utenti di computer perchè, ormai, nessuno più digita listati da riviste, ma preferisce procurarsi i programi direttamente su nastro o disco rintracciabili a prezzo, ormai, molto contenuto.

Non tutti i listati che pubblichiamo devono, necessariamente, esser digitati ma, soprattutto, studiati per estrapolarne la parte che interessa. Questo vale anche per coloro che possiedono una stampante e che potrebbero provvedere a riversare su carta i programmi che acquistano su supporto magnetico (ma quelli in circolazione sono tutti listabili?).

Molti lettori, inoltre, confessano che non digitano tutti i listati, ma che conservano gli arretrati della nostra rivista perchè un certo programma, in futuro, potrebbe tornare utile.

Non dimentichiamo che la nostra pubblicazione non vuole entrare in competizione con le riviste su supporto magnetico nè, tantomeno, pretende di presenatare giochi più spettacolari di quelli pirateggiati, ci mancherebbe altro!

Noi ci rivolgiamo a coloro, e sono tanti, che desiderano imparare a programmare, a prender confidenza con il computer e, credetemi, nulla è più valido a tale scopo che l'esame del lavoro svolto da altri.

Va da sè che la qualità della carta usata, e la "veste" grafica, è da noi posta in secondo piano rispetto al contenuto della rivista stessa: sono convinto che ciò che conta non è la confezione nè il numero di "sedicesimi" in quadricromia. La carta patinata, semmai, può servire per nascondere contenuti non sempre utili o necessari e giustificare un prezzo di copertina più elevato.

Fai un esperimento: confrontando tra loro varie riviste di informatica, tra cui la nostra, prova a contare il numero (e la qualità) degli articoli che realmente ti hanno interessano al termine della lettura...

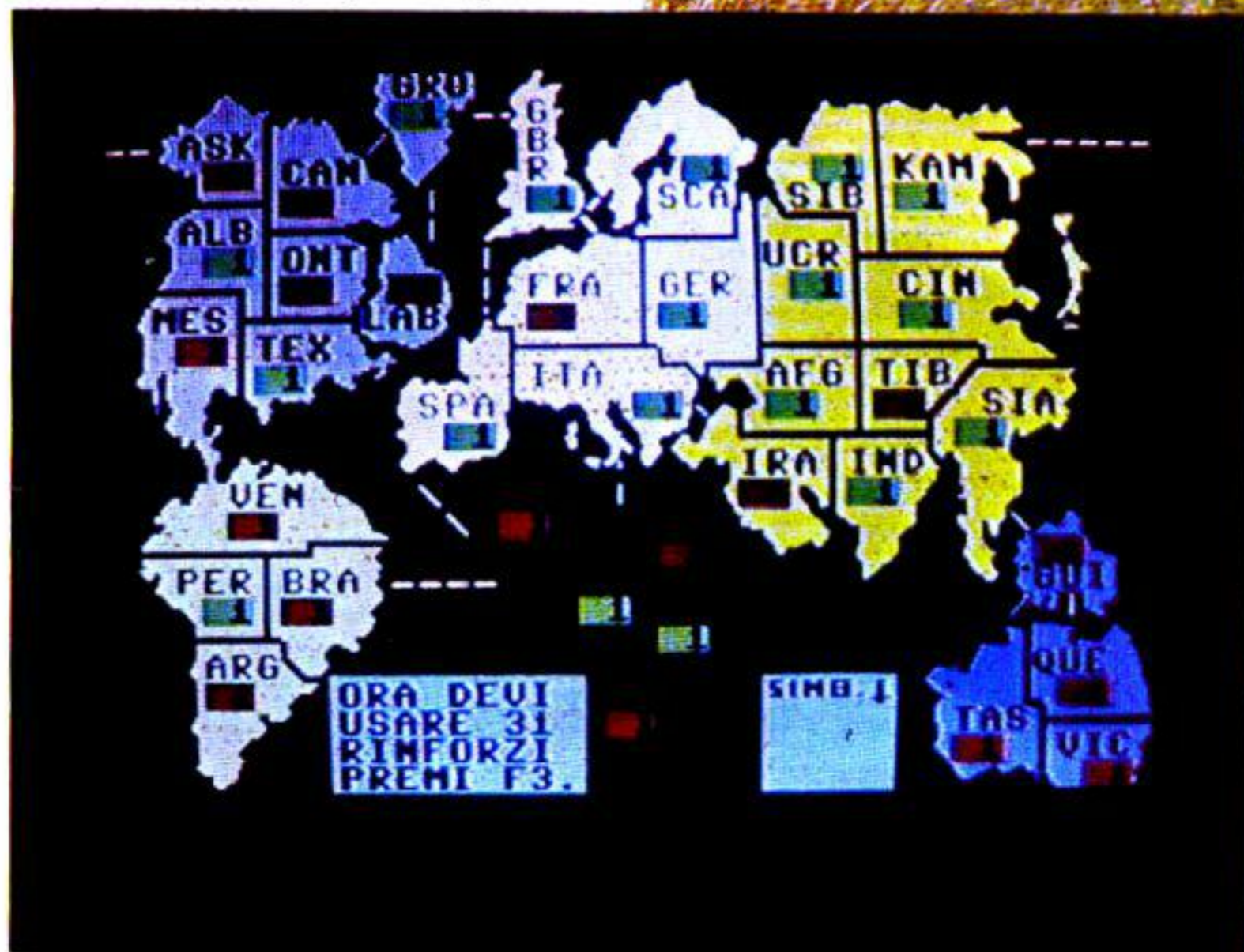
La guerra per gioco

Una panoramica sul contenuto del secondo numero della cassetta Commodore 64 Club

Questo secondo numero contiene, oltre ai consueti videogame ed utility, un fantastico programma di simulazione bellica ispirato al famosissimo Risiko, il popolare Wargame da tavolo.

Risicom 64

Dopo aver caricato e mandato in esecuzione il programma, dovrete digitare il numero dei giocatori, che varia da due a sei ed il vostro nome.

Successivamente inizieranno i "combattimenti" gestiti interamente dal computer che vi vedranno impegnati non solo in azioni strategiche e di ragionamento ma anche in veri e propri videogame.

Certamente la peculiarità di questo gioco risiede nel fatto che a momenti di freddo calcolo e di studiata simulazione si alternano momenti di puro gioco in cui, Joystick alla mano, dovrete impegnarvi in scontri con il nemico.

Alla cassetta è allegato un manuale esplicativo che ha lo scopo di elencare dettagliatamente tutte le modalità di gioco e di attacco nonchè tutte le divisioni territoriali necessarie al funzionamento del gioco.

Risicom 64 è senz'altro uno dei migliori prodotti software ORIGINALI mai presentati sul mercato italiano e certamente soddisferà appieno sia l'appassionato di Arcade-games sia chi, più "tradizionelista", ama le simulazioni a tavolino e predilige i giochi di ragionamento.

Mission Twain

Il secondo programma che vi presentiamo è una stupenda avventura grafica gestita in Linguaggio Macchina che vi farà viaggiare in fantastici mondi lontani anni-luce dalla Terra.

Il pallido sole del sistema Twain si alza lentamente all'orizzonte, liberando il lussureggiante mondo dall'oscura presenza delle ombre.

Ancora nessuno è riuscito a capire il motivo della presenza della vita su questo pianeta decisamente inospitale, eppure la foresta vergine si stende in tutte le direzioni, come un'immenso mare verde che si apre al tuo passaggio per poi chiudersi dietro di te.

Eppure il tuo assistente aveva avuto quella sensazione... di pericolo... o di premonizione? Non occorre scervellarsi; tu non sei in grado di pilotare l'astronave da solo ed i tuoi compagni sono spariti da ormai tre giorni... è assolutamente necessario ritrovarli prima che... prima che anche tu sparisca.

Il gioco Mission Twain è un'avventura ambientata nel prossimo futuro; la terra è alle prese con i problemi del sovraffollamento. Contemporaneamente si sviluppa notevolmente la scienza spaziale, che permette all'uomo di aprire nuovi orizzonti nei più nascosti angoli della galassia.

Il pianeta Twain si è rilevato un'ottima sorgente di materiale per quanto riguarda gli studi sull'ambiente e sull'adattamento umano ai diversi sistemi stellari; inoltre è quello che mostra il maggior numero di forme di vita nella galassia, oltre ovviamente alla terra. Il pianeta Twain è però anche un incredibile investimento nel campo minerario e molti interessi sono in gioco... forse troppi.

La relazione inviata dalla sonda automatica di Aldebaran pone però nuovi quesiti: forse il pianeta Twain è un luogo pericoloso ed inospitale; la tua spedizione dovrà provarlo... sempre se riuscirai a tornare!

Demoni

In questo gioco dalle stupende qualità grafiche e sonore, dovrai impersonare il mago Gillot che, impegnato nella eterna lotta contro le forze del male, scenderà negli abissi oscuri, nel regno di Terrorland.

Dopo anni di continui ed intensi studi, il mago Gillot si è reso conto che a nulla sarebbe servito cercare di combattere il nemico senza conoscerlo interiormente; così un bel giorno il mago Gillot decise di partire per Terrorland con il suo assistente Italus.

L'imprevisto però era dietro l'angolo; l'assistente Italus non fu insensibile alle lusinghe di Straplon, il re di Terrorland ed in una notte buia e malvagia Italus fuggì.

Il mattino mostrava a Gillot il grigiore di una potenza oscura proveniente dal centro di Terrorland, al quale lui si era incautamente avvicinato. Solo allora si rese conto della scomparsa di Italus; il terrore lo assalì e quando si rese conto di essere circondato dalle forze del male capì di avere ben poche possibilità di salvezza.

Cominciò così a correre verso la pace della sua casa ma la via era lunga e pericolosa e Straplon gli aveva inviato contro tutto il suo malefico esercito. No... non sarebbe stato facile sfuggire ai Demoni...

Level shock

In questo entusiasmante gioco dove la vostra abilità e prontezza di riflessi saranno messi a dura prova, impersonerete un avventuriero cacciatore di tombe rimasto imprigionato nel palazzo imperiale nelle rovine di Uruk.

L'antico architetto aveva evidentemente previsto una simile situazione e le trappole abbondavano dovunque; infernali marchingegni che muovono pesanti muri in tutte le direzioni. Riuscirai, utilizzando il tuo zainetto propulsore, a risalire i vari livelli della tomba ed a trovare l'uscita?

Errori made in Italy.

Capita abbastanza spesso che un programmatore alle prime armi si veda costretto a sfogliare voluminosi manuali di istruzioni per comprendere il significato di quegli incomprensibili messaggi inviati dal computer in seguito a probabili errori di programmazione.

Capita anche, forse un po' meno spesso, che alcuni programmatori meno pazienti decidano di interrompere la loro attività a causa delle difficoltà della lingua.

A questo punto entra in gioco la creatività dei nostri collaboratori ed ecco pronti... i messaggi di errore tradotti in italiano, ad uso e consumo di tutti coloro che non si sono ancora arresi. Vediamo quindi il fatidico "Syntax error" trasformato in un amichevole (e comprensibile!) "Errore di sintassi", un cattivo "Bad subscript error" corretto in "Matrice non definita" e così via, per la gioia di tutti coloro che si sono appena avvicinati allo sconfinato mondo del computer.

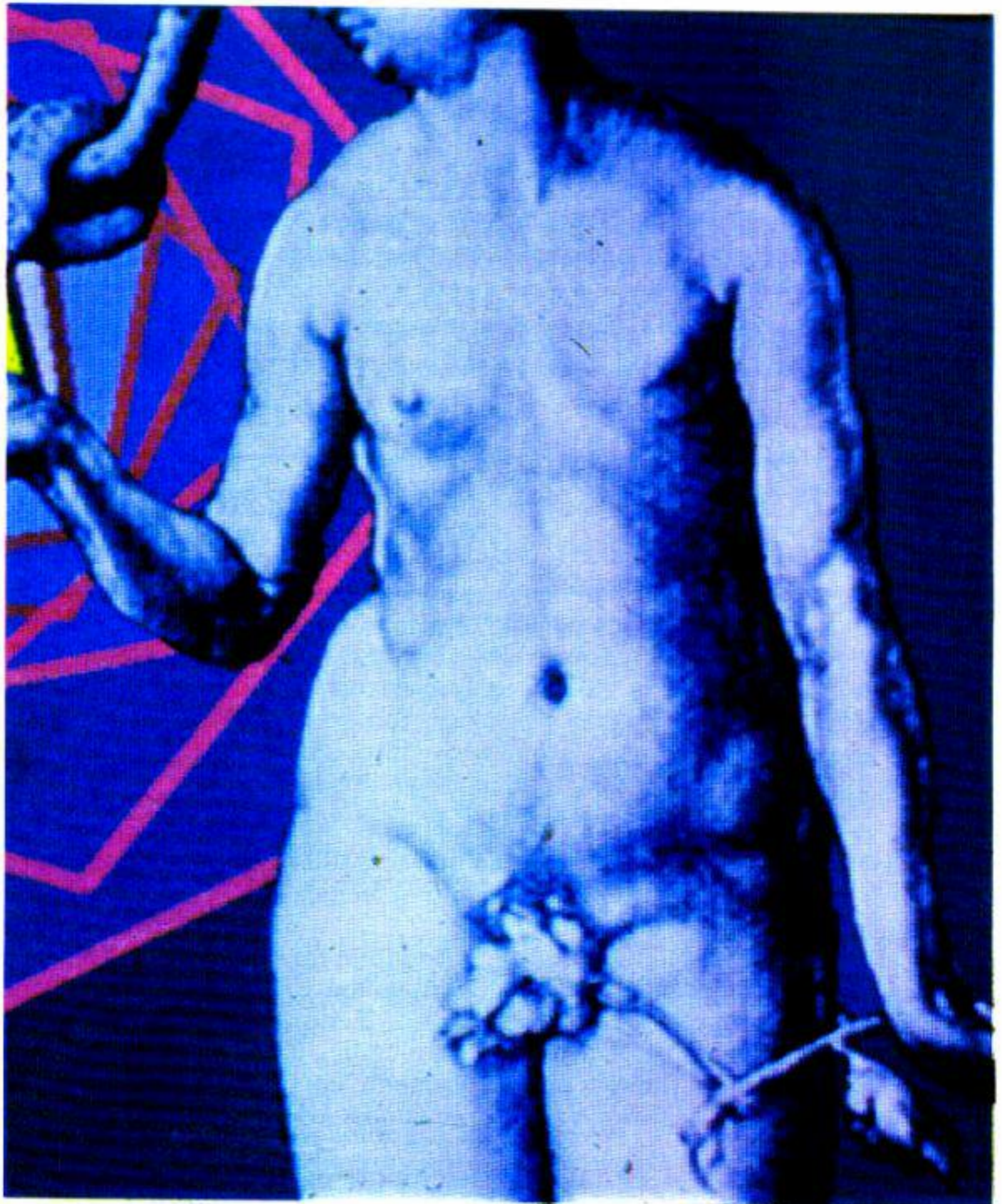
Ai confini della realtà.

Ore 23.19. Si incendia il motore che è stato appena colpito da un fulmine.

L'aereo perde quota e tu, giovane e brillante manager di successo, decidi di abbandonare l'aereo in modo da poterti salvare.

Mentre il paracadute si apre scorgi sopra di te l'aereo in fiamme che si allontana tra le nuvole basse e poi più nulla...

Un terribile colpo di vento spezza una delle cinghie del paracadute e tu piombi pesantemente nelle gelide acque dell'oceano atlantico; senti le gambe irrigidirsi ma riesci ugualmente a liberarti del paracadute; perdi i sensi...

Il raggio di sole ti colpisce gli occhi e tu ti risvegli su una assolata spiaggia... deserta, sembra. Dopo il primo attimo di smarrimento decidi di esplorare l'isola e cercare di capire dove ti potresti trovare. Probabilmente tu non capirai molto di navigazione e di mappe nautiche ma la presenza di un isola al centro dell'oceano atlantico è alquanto sospetta e non sapresti davvero darle un nome... non razionalmente almeno.

E' inutile però restare sulla spiaggia ad aspettare; è necessario invece cominciare ad esplorare l'isola e cercare qualche possibilità di salvezza.

Nazioni europee.

Il professore si siede dietro la cattedra e lentamente, quasi in modo solenne, apre il registro personale.

Il dito scorre nervosamente tra i nomi mentre la tensione cresce a dismisura tra le file dei banchi; tutti gli occhi sono puntati verso le labbra del professore che alla fine si muovono e pronunciano il cognome... .

L'interrogazione. L'atto che pone una solida e permanente barriera tra il professore e l'alunno il punto d'incontro degli sforzi trimestrali e forse di due punti di vista.

Ora il computer veste i panni di un preciso ed attento interrogatore che pone all'alunno diverse domande di tipo geografico riguardo le nazioni d'Europa; presentandone anche una rappresentazione grafica.

Dopo una serie di trentatrè domande il computer formula un giudizio in base alle risposte date ed assegna un voto allo studente. Buona fortuna... anzi, in bocca al lupo!

Gruppi di caratteri in un interno

Tre routine per la gestione finalizzata di vari gruppi di caratteri alfanumerici (e non)

17645 Simula Amiga menu
(Commodore 64)

Nell'Amigabasic è disponibile un comando comodissimo che permette di gestire efficacemente la tecnica delle finestre selezionabili da mouse.

Con il comando MENU ON, infatti, è possibile far apparire, sul primo rigo in alto, diverse parole-chiave, ognuna delle quali rappresenta altrettanti sub-menu.

Posizionandosi con il mouse su uno di tali codici, e clickando, compare la finestra con i sub-menu tra i quali scegliere quello desiderato sovrapponendosi ancora con il mouse.

Dietro la notevole semplicità d'uso descritta si nasconde, al contrario, una notevole complessità operativa: il computer, infatti, deve selezionare il mouse come unità di ingresso dati, individuarne la posizione e creare la finestra corrispondente definita in precedenza. Naturalmente la parte del video "nascosta" dalla finestra continua a esistere e riappare non appena la finestra viene eliminata.

La routine proposta non è tanto sofisticata, viene gestita da tastiera e non da joy (nè tantomeno da mouse) ma rappresenta un'idea da sviluppare per proprio conto.

Esaminando il listato notiamo che questo, in accordo con i canoni dell'Enciclopedia, è suddiviso in due parti; la prima è un programma dimostrativo (righe 140/190); la seconda è la subroutine vera e propria (17600/17699).

Prima di accedere alla subroutine 17600 è necessario assegnare ai quattro elementi del vettore stringa X0$() le quattro parole chiave che, qualunque sia la loro lunghezza, verranno ridefinite in seguito (riga 17600) con lunghezza fissa di 10 caratteri ciascuna, per questioni di simmetria.

Al vettore bidimensionale Y0(...,..) spetta invece il compito di contenere un certo numero di elementi che rappresentano, gruppo per gruppo, le varie opzioni corrispondenti a ciascuna parola chiave.

Nell'esempio dimostrativo sono stati assegnati quattro nomi di fantasia, ma modificando il valore della variabile X3 (riga 140) è possibile avere una certa libertà di scelta. Si tenga presente, però, che il video è formato da 25 righe e che le prime quattro sono "assorbite" dalla stessa subroutine per visualizzare efficacemente le parole chiave; si tenga anche presente che una qualsiasi matrice o vettore, utilizzato senza essere stato dapprima dimensionato (ed è proprio il caso dello standard adottato dalla nostra Enciclopedia) non può avere più di dieci elementi per ciascuna dimensione (numerati da 0 a 10), pena un Bad Subscript Error.

Dopo aver cancellato lo schermo, aver formattato le parole chiave e averle visualizzate in reverse, una linea tratteggiata separa la zona menu da quella sub-menu (righe 17600/17605).

L'ultima istruzione di 17605 esamina il tasto premuto che, ovviamente, deve essere 1, 2, 3 oppure 4 e lo memorizza nella variabile X6.

Il menu selezionato viene visualizzato in "normale" e il programma attende la conferma con la pressione del tasto Return (riga 17615). In caso contrario si riparte dall'inizio.

Premendo Return, quindi, viene visualizzato, in reverse, l'intero sub-menu relativo alla parola chiave, tra cui scegliere premendo il tasto numerico corrispondente. Anche in questo caso viene visualizzato in "normale" il sub menu selezionato e con la pressione del tasto Return si "esce" definitivamente dalla subroutine stessa.

Al ritorno, nel programma principale, la variabile X6

(oppure X5) conterrà il numero della parola-chiave, mentre X7 l'opzione corrispondente al sub-menu relativo.

Tali valori, ovviamente, devono essere opportunamente gestiti dall'utente per "saltare" ad altre parti del programma.

Si sottolinea che, nella riga 17600, gli spazi bianchi sono 10 mentre nella 17625 sono 20. L'adattabilità ad altri computer dipende unicamente dalla diversa formattazione del testo e dalle Poke di schermo (righe 17655 e 17660).

```
  100 REM DEMO SIMULATORE
  110 REM DI ISTRUZIONE MENU
  120 REM PER COMMODORE 64
  130 :
  140 X0$(1)="PRIMO":X0$(2)="SECO
      NDO":X0$(3)="TERZO":X0$(4)=
      "QUARTO":X3=4
  150 Y0$(1,1)="ALFA":Y0$(1,2)="B
      ETA":Y0$(1,3)="GAMMA":Y0$(1
      ,4)="DELTA"
  160 Y0$(2,1)="UNO":Y0$(2,2)="DU
      E":Y0$(2,3)="TRE":Y0$(2,4)=
      "QUATTRO"
  170 Y0$(3,1)="ONE":Y0$(3,2)="TW
      O":Y0$(3,3)="THREE":Y0$(3,4
      )="FOUR"
  180 GOSUB 17600:PRINTCHR$(147)"
      HAI SCELTO:":PRINT"SUB-MENU
      : "X0$(X6)
  190 PRINT"OPZIONE:   "Y0$(X5,X7)
 9998 :
 9999 END
17600 PRINTCHR$(147)CHR$(18);:FOR
       X1=1 TO 4:PRINTLEFT$(X0$(X
      1)+"          ",10);
17605 NEXT:PRINT:FOR X1=0 TO 39:P
      RINT"-";:NEXT:GOSUB 17645
17610 IF X4<1 OR X4>4 THEN 17600
17615 X6=X4:GOSUB 17655:X5=X4:GOS
      UB 17645:IF X0$<>CHR$(13) T
      HEN 17600
17620 PRINT"[HOME]":PRINT:FOR X2=
      1 TO X3
17625 PRINT"[RVS]"LEFT$(Y0$(X5,X2
      )+"                    ",20)
      :NEXT
17630 GOSUB 17645:IF X4<1 OR X4>X
      3 THEN 17600
17635 X7=X4:GOSUB 17660:GOSUB 176
      45:IF X0$<>CHR$(13) THEN 17
      600
17640 RETURN
17645 GET X0$:IF X0$="" THEN 1764
      5
17650 X4=VAL(X0$):RETURN
17655 FOR X2=1014+X4*10 TO 1014+X
      4*10+9:POKE X2,PEEK(X2) AND
       127:NEXT:RETURN
17660 FOR X2=1024+(3+X4)*40 TO 10
      24+(3+X4)*40+20:POKE X2,PEE
      K(X2) AND 127:NEXT:RETURN
17699 REM SIMULATORE ISTRUZIONE M
      ENU
```

17700 Memorizza messaggi
(Commodore 64)

E' noto che per memorizzare un gruppo di caratteri alfanumerici è sufficiente ricorrere alle stringhe e al comando Input. Molti sono, però, gli inconvenienti che un tale sistema comporta: anzitutto le stringhe, nonostante possano essere lunghe fino a 254 caratteri, hanno il limite "fisico" di 77 caratteri se associato a un Input. Provando, ad esempio, il semplice programmino...

1 Input A$: Print
2 Print A$: Goto 1

...possiamo notare che il primo carattere digitabile, dopo la comparsa del punto di domanda tipico dell'Input, si trova distante due spazi dal bordo video sinistro; l'ultimo carattere digitabile, invece, è quello che, al momento di battere il tasto Return, si trova sul rigo successivo a distanza di uno spazio dal bordo destro. In pratica, come potrete verificare voi stessi digitando la riga-programma di prima, una stringa non può essere più lunga di 77 caratteri.

Se ciò non bastasse non è possibile digitare il carattere di virgola (,) e doppio punto (:) che vengono interpretati dal computer come "fine" di un dato battuto.

La forma di una stringa digitata in fase di Input, infine, non è la stessa in fase di Print (provate a digitare una frase completa con il programma di prima).

Per memorizzare, e richiamare, senza problemi messaggi lunghi quanto si vuole, viene proposta la routine di queste pagine che memorizza, oppure richiama, un messaggio a seconda del contenuto della stringa X0$ (Store oppure Recall); righe 17700/703).

Nel primo caso lo schermo viene cancellato, le prime cinque righe (per un totale di 200 byte) visualizzate in reverse ed un messaggio invita a digitare il testo che può essere formato da uno qualunque dei caratteri assegnati alla stringa X1$ (riga 17710). In seguito, un semplice ciclo di attesa pressione tasto verifica che il tasto battuto corrisponda, o meno, ad uno di quelli leciti, provvedendo ad ignorare tutti gli altri.

La pressione del tasto Return provoca la richiesta del numero del banco (da 200 byte) che si intende riempire

con il messaggio digitato. La zona di memoria utilizzata è quella compresa tra 49152 e 53247 (4K) che, suddivisa idealmente in "banchi" da 200 byte ciascuno, consente, appunto, la memorizzazione di ben 20 messaggi (numerati da 0 a 19).

E' ovvio che in tale zona di memoria non devono esser presenti programmi in linguaggio macchina o altri oggetti informatici, che verrebbero irrimediabilmente distrutti.

Con un procedimento analogo è possibile richiamare uno qualunque dei banchi precedentemente memorizzati.

Il lettore può aumentare o diminuire a piacere la dimensione del banco tenendo presente che, per meno di 80 caratteri, è più conveniente il ricorso alle stringhe e che per oltre 500 caratteri conviene memorizzare l'intera pagina video.

```
  100 REM DEMO MEMORIZZA E RICHIA
      MA MESSAGGI PER C/64
  110 :
  120 INPUT "MEMORIZZI MESSAGGIO
      (S/N)";A$
  130 IF A$="S" THEN X0$="STORE":
      GOSUB 17700:GOTO 120
  133 INPUT "RICHIAMI MESSAGGIO (
      S/N)";A$:IF A$<>"S" THEN 12
      0
  140 INPUT "RICHIAMA BANCO (0/19
      )";X1
  150 IF X1<0 OR X>19 THEN 133
  160 X0$="RECALL":PRINTCHR$(147)
      :GOSUB 17700
  170 PRINTCHR$(19):FOR I=1 TO 6:
      PRINT:NEXT:GOTO 120
 9998 :
 9999 END
17700 IF X0$="STORE" THEN 17708
17703 IF X0$="RECALL" THEN 17770
17705 X0$="ERR":RETURN
17708 X1$=" 1234567890+-£QWERTYUI
      OP@*↑ASDFGHJKL:;=ZXCVBNM,./
      !#$%&'()[]<>?[LEFT][UP]"
17710 PRINTCHR$(147):FOR X0=1024
      TO 1024+199:POKE X0,PEEK(X0
      ) OR 128:NEXT:X0=LEN(X1$)
17720 PRINT"[5 DOWN]DIGITA IL MES
      SAGGIO[HOME]";
17730 GET X0$:IF X0$="" THEN 1773
      0
17732 X2=0:FOR X1=1 TO X0:IF MID$
      (X1$,X1,1)=X0$ THEN X2=1:X1
      =X0
17734 NEXT:IF X0$=CHR$(13) THEN 1
      7740
17735 IF X2=1 THEN PRINTX0$;
17736 GOTO 17730
17740 PRINT"[HOME][5 DOWN]"
17750 INPUT "N.BANCO DI MEMORIA (
      0/19)";X0:IF X0<0 OR X0>19
      THEN PRINT"[UP]";:GOTO 1775
      0
17760 X1=49152+X0*200:FOR X0=0 TO
       199:POKE X1+X0,PEEK(1024+X
      0):NEXT:RETURN
17770 FOR X0=0 TO 199:POKE 1024+X
      0,PEEK(49152+X1*200+X0):NEX
      T:RETURN
17790 REM X1 CONTIENE N.BANCO(IN
      RECALL)
17799 REM MEMORIZZA E RICHIAMA ME
      SSAGGI
```

17800 Legge file da disco
(Qualsiasi Commodore)

Anche questo terzo sottoprogramma gestisce gruppi di caratteri, ma stavolta si tratta di file registrati su disco.

Come è noto, i sistemi operativi più evoluti (tra cui quello dell'Amiga) hanno un comando specifico per esaminare i file presenti su disco.

Per il C/64 non è disponibile un'utility di questo tipo ma, con una manciata di righe Basic, è possibile supplire alla carenza lamentata.

Un file può essere di tipo sequenziale (s), programma (p), relativo (r) e user (u). Il più delle volte un file sequenziale è un file generato da un Word Processor e, come tale, formato da caratteri Ascii, stampabili senza ricorrere a "traduzioni" più o meno sofisticate.

I file programma, invece, contengono byte i cui valori sono tra i più vari e non visualizzabili mediante un semplice Print Chr$(X); il motivo è presto detto: molti codici di controllo del cursore (Up, Down, Clear, Home, Insert, Delete) e quelli relativi al colore o ai tasti funzione, hanno lo stesso valore di alcuni codici macchina, oppure di puntatori, di Token e di Link presenti nei programmi Basic. Tentando di estrarre da un file programma i vari codici di cui è costituito, e cercando di stamparli con il semplice ricorso a Chr$ si rischia, a parte la visualizzazione di caratteri privi di senso, di cambiare colore al cursore, di cancellare lo schermo e, nei casi più gravi, di premere Run/Stop e Restore per recuperare il controllo del computer.

La routine proposta, accessibile con un semplice Gosub 17800, chiede il nome e il tipo del file che si intende esaminare. Subito dopo apre due canali: il primo, di errore, e il secondo di lettura vera e propria.

Nel caso in cui il file non esista, oppure esiste ma non è del tipo indicato, il programma si interrompe e un messaggio opportuno segnala il tipo di errore incontrato.

In caso contrario, invece, il file sequenziale viene letto e visualizzato carattere per carattere, mentre degli altri tipi di file vengono visualizzati i vari codici (decimali) da cui sono costituiti.

Si avverte che con alcuni file sequenziali possono presentarsi egualmente gli inconvenienti prima segnalati, specie se tali file rappresentano schermate grafiche in alta risoluzione o, comunque, non file di testo.

Premendo un qualsiasi tasto durante la visualizzazione, si ritorna al programma principale, dopo aver chiuso i canali aperti in precedenza.

```
  100 REM DINMOSTRATIVO DI LETTUR
      A FILE SU DISCO
  110 GOSUB 17800
  120 :
 9999 END
17800 INPUT "NOME FILE";X0$:IF X0
      $="" OR LEN(X0$)>16 THEN X0
      $="ERR":RETURN
17802 INPUT "TIPO FILE (S,P,U,R)"
      ;X1$:IF X1$="" OR LEN(X1$)>
      1 THEN X0$="ERR":RETURN
17805 GOSUB 17880:OPEN 15,8,15:RE
      M APRE CANALE DI ERRORE
17806 OPEN 8,8,8,X0$+","+X1$+",R"
      :REM APRE FILE IN LETTURA
17810 GOSUB 17870:IF X0>0 THEN 17
      880:REM VERIFICA SE FILE ES
      ISTE
17820 GET #8,X0$:IF ST>0 THEN 178
      80:REM VERIFICA SE IL FILE
      NON E' FINITO
17825 GET X2$:IF X2$<>"" THEN 178
      80:REM ESAMINA EVENTUALE PR
      ESSIONE DI TASTO
17830 IF X1$="S" THEN PRINTX0$;:G
      OTO 17820:REM SE FILE SEQUE
      NZIALE STAMPA STRINGA-
17834 IF X0$<>"" THEN PRINTASC(X0
      $);:REM SE ALTRI FILE STAMP
      A CODICE ASCII
17836 GOTO 17820:REM RICOMINCIA
17870 INPUT#15,X0,X0$:IF X0=0 THE
      N X0$="OK":RETURN:REM LEGGE
       CANALE DI ERRORE
17872 PRINT"ERRORE"X0;X0$:X0$="ER
      R"
17880 CLOSE 8:CLOSE 15:RETURN
17899 REM LEGGE FILE DA DISCO
```

Elenco delle ultime routine pubblicate

(Fra parentesi è riportato il numero di Commodore Computer Club su cui sono apparse)

63913 rem 17500 Mini Text ed. (42)
63914 rem 17400 Hard Copy L/R (42)
63915 rem 17300 Grandezze Anal. (41)
63916 rem 17200 Interp a$ (41)
63917 rem 17100 equivalenze (40)
63918 rem 17000 percentuali (40)
63919 rem 16900 deek & doke (39)
63920 rem 16800 sprite scanner (39)
63921 rem 16700 movimento sprite (39)
63922 rem 16600 accensione sprite (39)
63923 rem 16500 drum per c/64 (38)
63924 rem 16400 draw low/res (38)
63925 rem 16300 print v/cont (38)
63926 rem 16200 plot low-res (37)
63927 rem 16100 integrali (37)
63928 rem 16000 equaz. mista (37)
63929 rem 15900 equaz. terzo gr. (37)
63930 rem 15800 derivata di funz. (37)
63931 rem 15700 scritte rotanti (37)
63932 rem 15600 convers. coordin. (36)
63933 rem 15500 logar. base quals. (36)
63934 rem 15400 conversione basi (36)
63935 rem 15300 semplif. frazioni (36)
63936 rem 15200 divis. con N decim.(36)
63937 rem 50100 directory (35)
63938 rem 15100 lampeggio righe (35)
63939 rem 15000 frammenta schermo (35)
63940 rem 14900 delete window (35)
63941 rem 14800 cambia stringhe (34)
63942 rem 14700 slitta stringhe (34)
63943 rem 14600 ruota stringhe (34)
63944 rem 10500 input programmab. (34)
63945 rem 14500 scroll solo testo (33)
63946 rem 14400 sprite multiuso (33)
63947 rem 14300 zoom esadecimale (33)
63948 rem 14200 video orologio (33)
63949 rem 11100 funzioni inverse (32)
63950 rem 13200 centra messaggi (32)
63951 rem 14100 finestre di testo (32)
63952 rem 14000 gestione nome disk (32)
63953 rem 13900 load/save pg.video (31)
63954 rem 13800 scritte in ebcm (31)
63955 rem 13700 bit image mps/803 (31)
63956 rem 13600 or esclusivo (31)
63957 rem 13500 comandi extra prg (31)
63958 rem 13400 linee low-res. (31)
63959 rem 13300 elabora stringhe (31)
63960 rem 13200 centratura frase (32)
63961 rem 13100 menu con joy (30)
63962 rem 13000 menu con cursore (30)
63963 rem 12900 frase lampeggiante (29)
63964 rem 12800 bordo technicolor (29)
63965 rem 12700 fill memoria ram (29)
63966 rem 12600 text copy mps 803 (29)
63967 rem 12500 colore pag.testo (29)
63968 rem 12400 print using (31)
63968 rem 12400 print using (29)

Un modem Commodore omologato Sip

Notizie fresche fresche dal mondo Commodore: un accordo tra la multinazionale americana e la nostrana società dei telefoni

Il mondo della trasmissione dati ha da sempre affascinato i nostri lettori (e, ovviamente, anche noi).

Da molto tempo, infatti, riceviamo richieste di chiarimenti su come effettuare trasmissioni di dati servendosi di un "semplice" Commodore 64.

E tutte le volte, nostro malgrado, abbiamo cercato di rinviare la trattazione dell'argomento. Forse sapevamo qualcosa di ciò che si apparecchiava in Commodore e aspettavamo il momento proprizio per parlarne diffusamente? Chissà...

Ma bando alle ciance e passiamo alla divulgazione della notizia ufficiale, appena giunta in Redazione:

La Commodore Italiana, grazie ad un particolare accordo recentemente siglato con la Sip e la Seat, propone ai suoi fedeli utenti un modem (dal codice 64/99), omologato regolarmente dal Ministero delle Poste, da collegare ad un C/64 per consentire l'invio e la ricezione di dati.

In vendita saranno poste particolari confezioni destinate sia a coloro che non posseggono il popolare computer (e che potrebbero approfittare dell'offerta per entrare nel mondo dell'informatica), sia a coloro che, possedendo un C/64, desiderano entrare in posesso di drive e del modem.

Confezione "Telematica 1"

- *Commodore 64*
- *Registratore 1530*
- *Geos (da usare, però, con il 1541)*
- *Modem 64/99*

Prezzo L.499.000 +IVA

Confezione Telematica 2

- *Drive 1541*
- *Mouse*
- *Modem 64/99*

Prezzo L.499.000 +IVA

Tali confezioni dovrebbero esser poste in vendita agli inizi di giugno e forse in questo momento qualche negoziante lo ha già posto in vetrina evidenziandolo degnamente.

Naturalmente, ma solo dal prossimo settembre, sarà possibile procurarsi il solo modem al prezzo di L.***150000*** al pubblico.

Per il momento ci limitiamo a divulgare la notizia e a pubblicare l'immagine del prototipo, così come è giunto in redazione.

Ci ripromettiamo di parlarne prossimamente, soprattutto per ciò che riguarda la possibilità di collegarsi gratuitamente con il Videotel e le Pagine Gialle Elettroniche.

Sembra che la Commodore abbia molta fiducia nel progetto: si prevede, nel solo mese di giugno, la vendita di non meno di 40000 (quarantamila) confezioni!

Considerando che i numeri 1, 2 e 7 sono esauriti, vogliate inviarmi i numeri arretrati .... al prezzo di L. 5.000 cadauno per richieste fino a 4 numeri, o di L. 4.000 cadauno per richieste oltre i 4 numeri arretrati, e perciò per un totale di L.............. Sono a conoscenza che i fascioli suddetti non saranno inviati in contrassegno e, pertanto, ho provveduto oggi stesso a versare il canone di L.................. a mezzo c/c postale n. 37952207 intestato a:
Systems Editoriale - V.le Famagosta, 75 - 20142 Milano

GIUDIZIO SUI PROGRAMMI DI QUESTO NUMERO

Ho assegnato un voto da 0 a 10 ai programmi che indico di seguito:

A/ .... Voto ....

B/ .... Voto ....

C/ .... Voto ....

D/ .... Voto ....

PICCOLI ANNUNCI ....

....

....

....

....

....

CERCO/OFFRO CONSULENZA ....

....

....

....

....

....

INVIARE IN BUSTA CHIUSA E AFFRANCANDO SECONDO LE TARIFFE VIGENTI A:

COMMODORE COMPUTER CLUB

V.le Famagosta, 75
20142 Milano

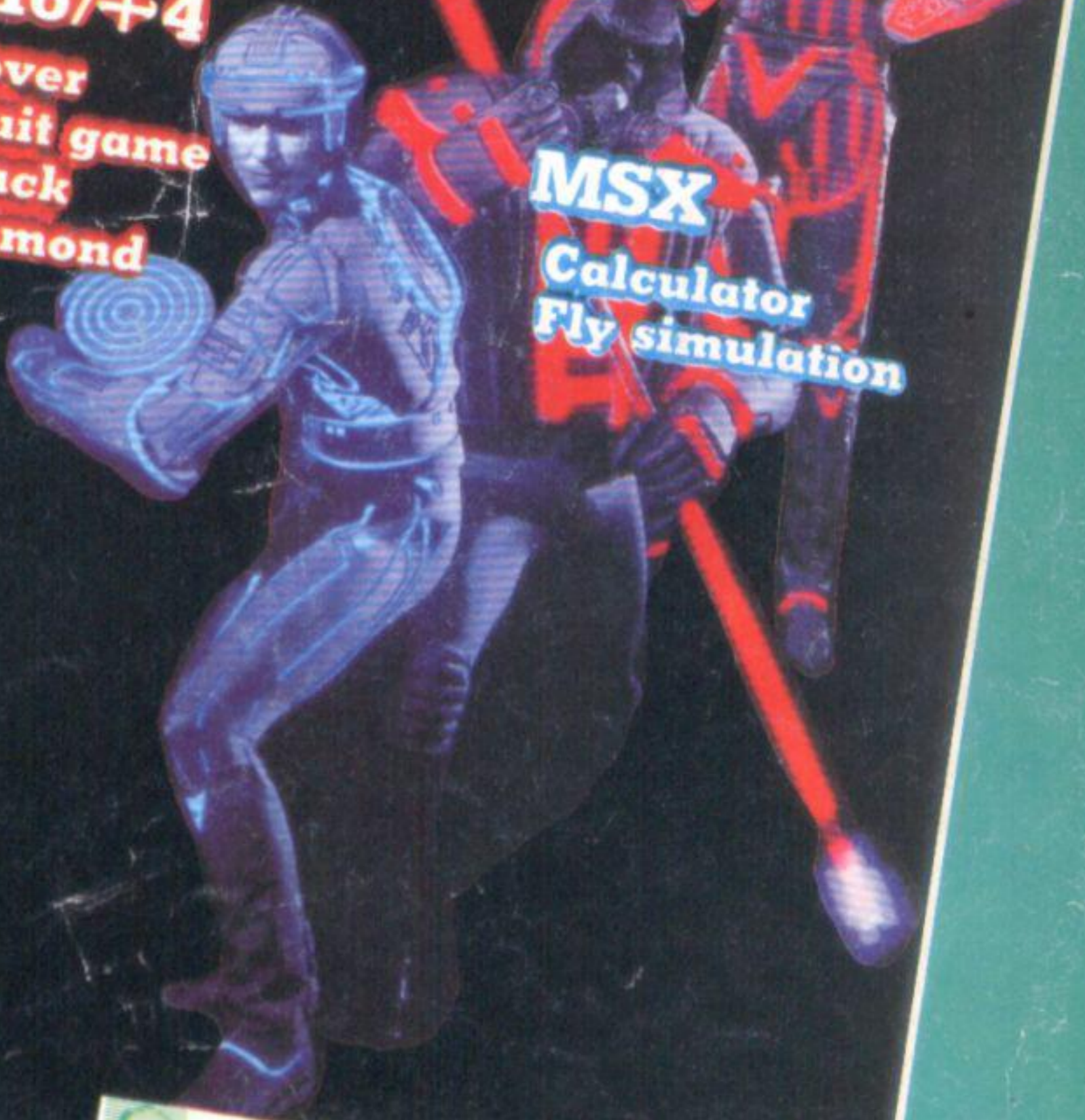
INVIARE TUTTA LA PAGINA ANCHE SE SI UTILIZZA UNA SOLA SCHEDA

Nome .... **Cognome** ....

Via .... **N°** .... **CAP.** .... **Città** ....

Telefono .... **Orario** ....

Quale fascicolo manca alla tua enciclopedia Commodore?

Per ordinare i fascicoli mancanti alla tua collezione di Commodore Computer Club utilizza l'apposita scheda in fondo alla rivista.

Viaggio nel Basic

Siete sicuri di conoscere come lavora il Basic durante la dichiarazione delle variabili, la stesura di un programma, la cancellazione di linee? Due inserti appositamente dedicati all'esplorazione dei meandri del vostro computer

di Alessandro de Simone

In queste righe illustreremo in modo approfondito la gestione delle variabili nei calcolatori Commodore servendoci di un gruppo di brevi programmi semplici e di immediata comprensione utilizzabili sul Commodore 64 (e sul Vic 20) e, con minime modifiche, anche sul C/16 e Plus/4.

Teniamo a precisare che gli argomenti affrontati possono sembrare molto difficili da comprendere.

Se, però, leggerete queste pagine avendo al fianco il computer acceso e digitando, volta per volta, quanto suggerito, avrete la possibilità di incrementare le vostre conoscenze nel campo della programmazione.

Organizzarsi

Dovrebbe esser noto che la memoria di un computer è una successione di byte che, nel caso del C/64, sono in quantità di 65536. Di questa gran massa di dati alcuni sono "fissi", vale a dire che, essendo su Rom, hanno il proprio valore ben definito e non possono esser modificati in alcun modo; per analogia possiamo affermare che la Rom corrisponde alla parte stampata di un'agenda.

La memoria Ram, al contrario, può essere scritta e cancellata a volontà dall'utente, o dallo stesso computer, e corrisponde, seguendo l'analogia precedente, alle pagine bianche della stessa agenda presa in considerazione in cui possiamo scrivere, cancellare, modificare qualsiasi messaggio, a patto, ovviamente, che ci serviamo di una matita (ed, eventualmente, di una gomma).

Come in un'agenda, a differenza di un qualsiasi block notes, ogni pagina rappresenta un giorno dell'anno e, comunque, un riferimento inconfondibile, così in un computer non tutte le locazioni hanno lo stesso valore e importanza.

Sappiamo benissimo, ad esempio, che un certo numero, posto nella rubrica telefonica, rappresenta il prefisso di un numero di telefono perchè è trascritto in una "cella" specifica. Lo stesso numero, posto altrove, può avere un significato totalmente diverso. La confusione, nel caso di un calcolatore, è decisamente maggiore perchè i numeri che è possibile memorizzare sono "soltanto" 256. Oltre a verificare il valore di una cella, quindi, è indispensabile conoscere con precisione la sua

posizione all'interno della memoria e, molto spesso, anche il valore delle celle adiacenti.

Se, ad esempio, digitate Print Chr$(65) otterrete la visualizzazione di una "A" maiuscola e ne potete dedurre che il numero 65 genera un carattere. Se, però, digitate Print Tab(65) sembra di non ottenere un bel niente perchè il computer si predispone a stampare qualcosa (che però non è indicato) a distanza di 65 celle video.

L'esempio appena riportato è volutamente banale, ma efficace per comprendere la differenza notevole di significato di uno stesso numero in circostanze diverse.

Abbiamo detto che in una memoria Rom (Read Only Memory, vale a dire, appunto, memoria a sola lettura) non è possibile scrivere nulla, ma solo leggerne il contenuto. Le Rom contengono, quindi, programmi scritti e memorizzati permanentemente dal fabbricante (la Commodore, nel nostro caso) idonei a far funzionare correttamente il calcolatore su cui le stesse Rom sono montate.

Una parte delle Rom consente, tra l'altro, di caricare, cancellare, scrivere, modificare programmi: sono in grado, insomma, di "capire" che stiamo digitando qualcosa sulla tastiera e di interpretare la nostra volontà quando battiamo List, Run, Save e in tutte le altre circostanze ben note.

Naturalmente il computer, grazie ai programmi permanenti scritti su Rom, non memorizza a casaccio i tasti che battiamo, ma provvede a inserirli razionalmente in aree di memoria Ram ben precise e in accordo ad una rigida architettura. Il nome Ram, che vuol dire "Random Access Memory" (memoria ad accesso casuale), non vuol dire che i dati vengono memorizzati a casaccio, ma che è possibile memorizzarvi un dato in una parte qualunque della Ram, senza seguire un percorso preciso. Ad esempio possiamo memorizzare un dato nella cella 9870 e, subito dopo, un dato nella 567 che non ha nulla a che fare con la precedente.

Il computer (o meglio: le Rom del Basic) hanno un orientamento molto valido per individuare il punto in cui sono memorizzati i dati che interessano nel corso di un'elaborazione. Anzitutto il computer "sa" che la zona Ram che può ospitare un programma Basic da noi scritto è circoscritta e ben delimitata; nel caso del C/64 i byte che possono ospitare un programma Basic sono quelli compresi tra 2049 e 40960 (per il C/16 il valore iniziale è 4096). Nella stessa area trovano posto

anche le variabili intere. in virgola mobile e stringa; e anche i vettori o le eventuali matrici. Con quale criterio il computer memorizza una riga Basic, una variabile, un vettore, una stringa? E, soprattutto, come fa a rintracciarli in seguito?

I puntatori

Prendiamo in considerazione l'abitazione di un nostro amico: per individuarla abbiamo bisogno di conoscerne l'esatto indirizzo (via e numero civico), ma tali informazioni, da sole, non bastano perchè il piano dell'edificio è altrettanto importante per recarci nel suo alloggio.

Allo stesso modo non serve a nulla sapere che abita al terzo piano se non conosciamo la via in cui abita.

Quando ci rechiamo da lui, quindi, "puntiamo" alla via e, individuato il portone d'ingresso, "puntiamo" al piano sul quale è ubicato il suo appartamento. Come si può notare, abbiamo bisogno di due informazioni che si rivelano indispensabili per rintracciare il nostro amico (e non altri). A pensarci bene, l'indirizzo è costituito non da uno, ma da due puntatori: il nome della via e il suo numero civico; allo stesso modo il puntatore del piano lo possiamo considerare costituito da due informazioni: il numero del piano e il numero della porta sullo stesso piano (la seconda da destra, la prima vicino all'ascensore, la terza in senso orario a partire dalle scale, eccetera).

Anche un computer, per individuare correttamente una locazione (o un gruppo di informazioni) necessita di puntatori, vale a dire di numeri che, opportunamente interpretati, gli consentono di accedere alle informazioni desiderate.

Gli indirizzi di inizio (2049) e di fine (40960) area destinata al Basic si possono ottenere da due puntatori che la individuano in qualsiasi momento.

Provate a digitare, qualunque sia il vostro computer:

*Print Peek(43)+ Peek(44)*256*
*Print Peek(55)+ Peek(56)*256*

Otterrete, nel caso di un C/64, i valori anzidetti, a patto che abbiate appena acceso il computer e che non abbiate combinato operazioni "strane", simili a quelle che verranno tra breve descritte...

Possiamo dedurre, quindi, che due celle adiacenti (43 e 44) contengono altrettanti valori che, combinati opportunamente, rappresentano l'indirizzo di una cella particolare. A tale coppia di byte diamo il nome di "Puntatore".

Con il sistema descritto sarà possibile individuare una qualsiasi tra le 65536 celle disponibili con un C/64, e più in generale, con un computer a 8 bit.

Molto spesso al secondo puntatore (quello, per intenderci, con indirizzo più alto, esempio: 44) si assegna il nome di "pagina" ed il computer, in questi casi, viene suddiviso in 256 pagine (numerate da 0 a 255); al primo puntatore (quello con il numero più basso: 43) è, invece, riservato il compito di individuare un byte all'interno della pagina indicata dal secondo puntatore. Naturalmente ogni pagina è costituita da 256 byte, numerati anch'essi da 0 a 255. Ecco spiegato, quindi, il valore 65536: non è altro che il prodotto di 256 (agine) x 256 (locazioni).

Elenchiamo, ora, i puntatori più importanti che si incontrano lavorando in Basic:

43/44: inizio del Basic
45/46: fine del Basic (e inizio variabili)
47/48: fine delle variabili (e inizio dei vettori e matrici)
49/50: fine dei vettori e delle variabili
51/52: primo byte in cui verrà depositata la prossima stringa
53/54: utilità
55/56: ultimo byte usato dal Basic

Riferendosi al C/64 potremo notare che, non appena accendiamo il computer, il contenuto dei puntatori, visualizzabile inserendo tra le parentesi di un banale Print Peek() la locazione desiderata, è il seguente:

*43/44: 1,8 (1+8*256=2049)*
*45/46: 3,8 (3+8*256=2051)*
47/48: 3,8
49/50: 3,8
*51/52: 0,160 (0+160*256=40960)*
53/54: 0,0
55/56: 0,160

Si noti che l'inizio del programma Basic (che... non c'è dal momento che abbiamo appena acceso il computer) è distante

tre byte dalla sua fine (2049 2051): l'area di inizio (45/46) e fine variabili e di inizio (47/48) e fine (49/50) vettori coincidono tra loro (2051).

Tale suddivisione, in parte contraddittoria, verrà studiata ampiamente in seguito. Per ora interessa esaminare in dettaglio ciò che succede quando iniziamo a digitare un qualsiasi programma. Trascrivete, ad esempio, le due righe seguenti:

100 REM ABC
110 REM ABC

facendo attenzione a lasciare un solo spazio tra le Rem e il gruppo di caratteri ABC e, soprattutto, a premere il tasto Return alla fine di ogni riga.

Se avete seguito alla lettera quanto detto, dovreste ritrovarvi (dopo opportuni Print Peek..) una situazione del genere:

*43/44: 1,8 (1+8*256=2049)*
*45/46: 23,8 (23+8*256=2071)*
47/48: 23,8
49/50: 23,8
*51/52: 0,160 (0+160*256=40960)*
53/54: 0,0
55/56: 0,160

Alcuni puntatori sono rimasti invariati (43/44, 51...56); gli altri, invece, sono cambiati proprio a causa della digitazione delle due righe Basic. Se i valori che riscontrate sono diversi, vuol dire che avete inserito più spazi oppure più righe o un numero diverso di caratteri alfanumerici dopo le Rem (oppure che non avete un C/64!).

Digitate ora di seguito la seguente riga, in modo diretto:

FOR I=2049 TO 2070: PRINT PEEK(I);: NEXT

...non trascurando di digitare il carattere di punto e virgola (;) dopo il Peek. Tale comando visualizzerà il contenuto (Peek) dei 21 byte numerati da 2049 a 2070. Se avete digitato senza errori, dovrebbe apparire:

11, 8, 100, 0, 143, 32, 65, 66, 67, 0
21, 8, 110, 0, 143, 32, 65, 66, 67, 0
0, 0

In effetti i valori compaiono tutti di seguito e la suddivisione in tre righe, che vedete in questa pagina, è riportata solo per motivi di chiarezza.

Noterete, infatti, che i primi due gruppi di dieci valori sono quasi simili tra loro; è poi riportata una coppia di zeri.

In particolare rileviamo che i primi due valori (11 e 8) "tradotti" con il solito sistema (11+8*256) danno come risultato il valore 2059; questo, a sua volta, "punta" alla locazione di memoria che contiene il valore 21.

Se, infatti, 11 è il contenuto della locazione 2049, 8 di 2050, 100 di 2051, si perviene a quanto asserito.

I valori 21 e 8, seguendo la stessa "traduzione", indicano la cella 2069 che contiene uno zero; stavolta, però, la coppia di zeri (2069/2070) "puntano" alla locazione zero (0+0*256=0), e tale particolarità fa capire al Basic che il programma è finito.

Il sistema appena esaminato è formato da una successione di puntatori "in cascata" detti, più propriamente, "Link" (=legame).

E', questo, un sistema semplice ed efficace per "inseguire" le informazioni che possono risultare disseminate in qualsiasi modo all'interno del computer; è una situazione simile a quella che si verifica in una caccia al tesoro, in cui ciascun luogo da rintracciare contiene un foglietto che, a sua volta, contiene le indicazioni idonee per rintracciare il successivo... e così via, fino al tesoro.

Ma esaminiamo meglio il significato dei 21 byte visualizzati: subito dopo il byte 2049 e 2050 (i link di prossima linea sui quali ci siamo già soffermati) sono presenti due byte (2051 e 2052) che contengono, rispettivamente, 100 e 0. Tale coppia di byte (successiva, cioè, ai due byte di Link) rappresenta la numerazione della linea Basic: 100+0*256=100. Con il numero 100, infatti, abbiamo numerato la prima riga del nostro microprogramma. Analogamente (vedi byte 2061/2062) la coppia di valori 110/0 indica la numerazione della seconda linea Basic.

Il valore 143 rappresenta, nel codice Commodore, il comando REM; il codice 32 è lo spazio e i valori 65, 66, 67, rispettivamente, i caratteri "a", "b" e "c".

Proviamo a modificare, prestando la massima attenzione, i contenuti di alcune locazioni.

In 2053 è presente il codice 143. Provate a digitare...

Poke 2053,153

...e, subito dopo, chiedete il listato: otterrete:

100 PRINT ABC
110 REM ABC

Il valore 153, infatti, è il codice Commodore di Print. Divertitevi a digitare altri valori (compresi tra 0 e 255) e, subito dopo, a richiedere il listato: in alcuni casi otterrete autentiche sorprese (scomparsa dei caratteri ABC, comparsa di segni semigrafici, istruzioni tipiche del Basic e così via). Alcune Poke hanno consentito, nel passato, di realizzare particolari tipi di protezioni.

Provate, ora, ad alterare il contenuto della locazione 2051 digitando, ad esempio...

Poke 2051,91

...e a chiedere il lsitato:

91 REM ABC
110 REM ABC

Ciò dimostra che, in effetti, la locazione 2051 è preposta (insieme con la successiva, la 2052) a contenere la numerazione della prima linea.

Divertendovi ad alterare, sempre mediante Poke, le locazioni relative alla numerazione delle due righe, otterrete risultati stranissimi tra cui la visualizzazione di due righe in ordine decrescente anzichè crescente.

Naturalmente alcune modifiche possono portare a veri e propri inchiodamenti della macchina che deve, in casi come questo, essere spenta e riaccesa per ripristinare le condizioni iniziali.

Nell'area destinata ad un programma Basic, insomma, è presente un doppio sistema di puntatori: una prima coppia (Link) che consente di conoscere la locazione Ram in cui è allocato il successivo Link; e una seconda coppia relativa alla numerazione Basic vera e propria. Ogni linea Basic termina con uno zero mentre un programma Basic è individuato da tre zeri posti in successione. Ma su questo argomento torneremo in dettaglio in seguito.

Per ora ci accontentiamo di affermare che l'interprete Basic del computer "sa" che nella locazione 2048 (puntata dalla coppia 43/44) è situato il primo dei due byte che contengono le informazioni necessarie per individuare l'intera area Basic, grazie alla "cascata" di puntatori posti in successione, finchè non incontra due zeri di seguito.

Esame dei puntatori

Abbiamo concluso il paragrafo precedente affermando che i byte 43 e 44 "puntano" all'inizio del programma Basic (se c'è) mentre i byte 45 e 46 puntano alla sua fine. In effetti i byte 45 e 46 puntano all'inizio delle variabili, ma, dato che queste sono allocate a partire dal byte successivo all'ultima locazione Basic, offrono comunque una valida indicazione per individuare il termine di un programma.

```
100 REM  LISTATO N.1
110 :
120 REM STUDIO DEI PUNTATORI:
130 REM QUANTITA' DI MEMORIA OCCUPATA DAL
140 REM BASIC E DA DIVERSE VARIABILI.
150 REM OPEN1,4:CMD1:REM PER STAMPARE I RISULTAT
    I
160 A=12:A$=""+"1234567890":A%=123
170 DIM A$(19):FOR A=0 TO 19:A$(A)=A$:NEXT
180 DIM A(19):FOR A=0 TO 19:A(A)=A:NEXT
190 DIM A%(19):FOR A=0 TO 19:A%(A)=A:NEXT
200 PRINTCHR$(147)"INIZIO BASIC   "PEEK(43)+PEEK
    (44)*256
210 PRINT"FINE   BASIC    " PEEK(45)+PEEK(46)*256
    :PRINT
220 PRINT"INIZIO VARIABILI (VEDI FINE BASIC)"
230 PRINT"FINE DELLE VAR." PEEK(47)+PEEK(48)*256
    :PRINT
240 PRINT"INIZIO MATRICI (VEDI FINE VARIABILI)"
250 PRINT"FINE MATRICI    " PEEK(49)+PEEK(50)*256
260 PRINT"ALLOCAZ.(PROSSIMA STRINGA) DA:" PEEK(5
    1)+PEEK(52)*256
270 PRINT"FINE STRINGHE   " PEEK(53)+PEEK(54)*256
280 PRINT"FINE MEMORIA    " PEEK(55)+PEEK(56)*256
290 PRINT"RAM (EXTRA PRG.) OCCUP.DA STRINGHE";
300 PRINT(PEEK(55)+PEEK(56)*256)-(PEEK(53)+PEEK(
    54)*256)-1:PRINT
310 PRINT"1 VARIABILE INTERA, 1 IN VIRG.MOB"
320 PRINT"1 VARIABILE STRINGA DI 10 CARATTERI"
```

```
330 PRINT"1 VETTORE DI 20 VALORI INTERI ";
340 PRINT"(2*20+7)":REM  A. DE SIMONE
350 PRINT"1 VETTORE DI 20 VALORI DECIM.";
360 PRINT" (5*20+7)"
370 PRINT"1 VETTORE DI 20 STRINGHE (3*20+7)";
380 REM PRINT#1:CLOSE1:REM COMANDO PER STAMPANTE
```

Digitate il programma N.1 e dategli il RUN: dovrebbe apparire quanto segue:

inizio Basic 2049
fine Basic 3094

inizio variabili (vedi fine Basic)
fine delle var. 3115

inizio matrici (vedi fine variabili)
fine matrici 3336
allocaz.(prossima stringa) da: 40750
fine stringhe 40760
fine memoria 40960
ram (extra prg.) occup.da stringhe 199

1 variabile intera, 1 in virg.mob
1 variabile stringa di 10 caratteri
*1 vettore di 20 valori interi (2*20+7)*
*1 vettore di 20 valori decim. (5*20+7)*
*1 vettore di 20 stringhe (3*20+7)*

Esaminiamo ciò che è accaduto, non senza aver dato dapprima una definizione:

Diremo che una variabile è "dichiarata" quando viene nominata per la prima volta nel corso di un'elaborazione all'interno di un programma Basic. Ciò significa, che nel caso del microprogramma che segue,...

100 a=100: b=3.456: a$="prova"

...la prima variabile ad essere dichiarata è "A", la seconda "B" e la terza A$. Osserviamo, invece, il caso seguente:

100 a=100
110 gosub 200

```
120 b=456: c$="primo"
130 ...
140 ...
200 c=34.87: d$="secondo"
210 return
```

Apparentemente l'ordine di dichiarazione delle variabili sembra essere:

A, B, C$, C, D$

Seguendo, invece, la struttura LOGICA del programma, il computer incontra dapprima la variabile A, ma, subito dopo, grazie a Gosub 200, memorizza "C" e "D$". Solo al "ritorno" dalla subroutine incontra "B" e "C$".

Quanto detto giustifica il motivo per cui, nei listati pubblicati, si è preferito comunicare immediatamente, nelle primissime righe, i nomi delle variabili che saranno adoperate nel corso dell'elaborazione dei singoli programmi. Il Basic, come infatti vedremo, alloca l'una in coda all'altra le variabili che a mano a mano incontra.

Ma torniamo al programma 1, ed esaminiamolo nei dettagli:

Riga 160

Dichiarazioni variabili: si noti la variabile A$ realizzata servendosi di una concatenazione. Ciò serve per "costringere" il Basic ad allocare i byte costituenti la stringa stessa in una zona estranea all'area del programma Basic, come vedremo più avanti.

Righe 170/190

Dimensionamento (e riempimento) di tre vettori di variabili intere, virgola mobile e stringhe lunghi ciascuno 20 elementi (non si dimentichi infatti la posizione "zero").

Righe 200/370

Stampa dei risultati. La visualizzazione (riportata nella precedente tabella) evidenzia in modo chiaro lo spazio occupato dalle variabili all'interno della memoria RAM.

Si precisa che i risultati di figura si riferiscono al Commodore 64. I possessori di Vic 20, C/16 e Plus/4 noteranno altri valo-

ri. Ciò è dovuto al fatto che gli indirizzi di partenza dei due computer sono diversi. Gli stessi utenti del C/64 potranno notare valori differenti da quelli riportati in figura. Il motivo deve esser ricercato nel fatto che i valori indicati cambiano a seconda della lunghezza del programma Basic. E' infatti sufficiente che, nella trascrizione del programma, un solo byte venga digitato in più, o in meno, perchè la lunghezza cambi (caso dei REM non trascritti, spazi bianchi tra istruzioni, ecc.).

Proprio a tal proposito il lettore può verificare le differenze esistenti digitando un maggior numero di righe, variabili, matrici, stringhe: in seguito a ciascuna modifica apportata, e dopo aver dato il Run, si potranno notare varie cose interessanti. Ne accenniamo alcune:

- *Aumentando, o diminuendo, il numero di righe Basic, o alterando la loro lunghezza, l'inizio del Basic non varia mai.*
- *Apportando variazioni viene modificato, invece, l'indirizzo dell'ultimo byte Basic.*
- *Se viene variato il puntatore di fine Basic (a causa della variazione apportata al programma stesso), vengono modificati TUTTI gli altri puntatori; tale alterazione è esattamente eguale (in più o in meno) alla variazione della lunghezza del programma.*
- *Ogni variabile (intera, decimale, stringa) occupa SEMPRE sette byte. Si deduce che la differenza tra i valori dei puntatori di fine ed inizio variabili è sempre un multiplo di sette (oppure vale zero nel caso non siano state dichiarate variabili).*
- *I puntatori di inizio e fine stringhe coincidono nel caso in cui le stringhe dichiarate siano allocate all'interno dell'area del programma Basic. Se invece, come nel listato presentato, esse rappresentano il risultato di una elaborazione di stringhe (Left$, Right$, somme, eccetera), la differenza tra i puntatori coinciderà con il numero dei caratteri costituenti le stringhe dichiarate, come avremo modo di studiare nel paragrafo relativo all'Overlay.*
- *I vettori occupano uno spazio diverso a seconda della propria tipologia:*

Vettori di variabili intere

Ogni vettore occupa il numero di byte seguenti:
2 per il nome del vettore (o matrice pluridimensionale).
2 per indicare il numero di byte occupati dall'intero vettore.
1 per indicare il numero delle dimensioni (max=256).

2 per ciascuna dimensione: ognuna di tali coppie ha il compito di indicare il numero di valori occupati dalla dimensione interessata.

Per ogni vettore dichiarato di variabili intere si ha pertanto un minimo di sette byte (con funzioni di "indice") necessari, al Basic, per ottenere informazioni sul vettore stesso. Oltre a quelli esaminati bisogna, ovviamente, aggiungere due byte per la memorizzazione di ciascun valore del vettore. Come è noto con due byte è possibile memorizzare valori interi compresi tra -32768 e +32767.

Vettori, o matrici, di valori decimali

Il numero di byte di "indice" sono gli stessi di quelli delle matrici intere (minimo 7). Cambia il numero dei byte per ciascun valore decimale: cinque invece di due.

Vettori stringhe

Idem come sopra per i byte di indice. Il numero per ciascun elemento del vettore è fissato invece in tre: il primo indica il numero dei caratteri della stringa esaminata, gli altri due individuano l'indirizzo del primo byte in cui è allocato il primo di essi. A questi vanno aggiunti, o meno (vedi dopo), i byte che costituiscono effettivamente la stringa stessa.

Il lettore, per verificare quanto asserito, può modificare a piacimento le linee 160-190 (listato n.1) inserendo, o cancellando, linee, dichiarando altre variabili, dimensionando in modo vario più matrici. Dopo aver apportato la variazione, digitando RUN sarà facile effettuare un controllo sui mutamenti avvenuti, specialmente per ciò che riguarda gli indirizzi di inizio e fine Basic, variabili, matrici e stringhe.

Nel caso particolare del programma N.1, le tre variabili dichiarate (A, A$, A%) rendono, appunto, pari a 3*7=21 la zona RAM dedicata ad esse. Il vettore A$(19), d'altra parte, occupa 7+3*20=67 byte, mentre A(19) richiede 7+20*5=107 e A%(19) 7+20*2=47 locazioni di memoria per un totale di 221 byte. Tale valore, aggiunto all'indirizzo di inizio matrici, fornisce, appunto, il valore del puntatore di fine matrici.

Esame variazione puntatori

```
100 REM  LISTATO N.2
110 :
120 REM ALLOCAZIONE DELLE VARIABILI NUMERICHE
130 REM PRIMA FASE: ESAME PUNTATORI DEL BASIC E
    DELLE
140 REM VARIABILI NEL CASO DI 6 DICHIARAZIONI
150 :
160 PRINTCHR$(14)CHR$(147)"PRIMA DI DICHIARAZION
    I":GOSUB  380
170 AA%=100:BB%=32767
180 PRINT:PRINT"DOPO DUE DICH."
190 GOSUB 380:PI=PI:PF=PF:I=I:J=J
200 PRINT:PRINT"DOPO 6 DICHIARAZIONI":GOSUB 380
210 PRINT:PRINT"DOPO MODIFICA A 2 DELLE 6 DICHIA
    RAZ."
220 AA%=111:BB%=-546:GOSUB 380
230 PI=PEEK(45)+PEEK(46)*256
240 PF=PEEK(47)+PEEK(48)*256
250 PRINT
260 PRINTCHR$(18)"ELENCO VARIABILI:"
270 :
280 FOR I=PI TO PF-7 STEP 7:PRINTCHR$(18);
290 PRINTCHR$(PEEK (I)) CHR$(PEEK(I+1))CHR$(146)
    "(";
300 PRINT PEEK(I) PEEK(I+1)")";
310 PRINT " "CHR$(18)I
320 FOR J=I TO I+6:PRINTPEEK(J);:NEXTJ
330 PRINT:NEXTI
340 PRINT"VALORI DELLE VARIABILI:"
350 PRINT:PRINT"AA%:"AA%;" BB%:"BB%;"  PI:"PI;"
     PF:"PF"  I:"I"  J:"J
360 END
370 REM SUBROUTINE DI ESAME PUNTATORI
380 PRINTCHR$(18)"INIZIO DEL BASIC" PEEK(43)+PEE
    K(44)*256
390 PRINT"FINE BA.-IN.VAR." PEEK(45)+PEEK(46)*25
    6
400 PRINT"FINE VARIABILI  " PEEK(47)+PEEK(48)*25
    6
410 PRINT"N.  VARIABILI  DICHIAR.=";
420 PRINT((PEEK(47)+PEEK(48)*256)-(PEEK(45)+PEEK
    (46)*256))/7
430 PRINT"N.BYTE PER VAR.DICHIAR.=";
440 PRINT(PEEK(47)+PEEK(48)*256)-(PEEK(45)+PEEK(
    46)*256)
450 PRINT TAB(25)CHR$(18)"PREMI UN TASTO"
```

```
460 IF PEEK(197)=64 THEN 460
470 RETURN
480 END
```

Il secondo listato presentato ha lo scopo di dimostrare che i puntatori del Basic vengono alterati anche *DURANTE* l'elaborazione dello stesso programma. Si noti infatti la subroutine 380-470: questa, quando viene richiamata, visualizza su schermo i valori dei puntatori che sono attivi in quel particolare momento dell'elaborazione.

Si noti, inoltre, che per il calcolo si è evitato il ricorso a variabili, proprio per rendere più comprensibile il listato stesso. Esaminiamolo, anzi, in dettaglio:

Riga 160

Prima che una qualsiasi variabile venga dichiarata, si utilizza la subroutine 380. Il risultato dimostra che, almeno in questa prima fase, i puntatori di inizio e fine variabili coincidono fra loro, proprio perchè non ne sono state dichiarate.

Riga 170

Vengono dichiarate due sole variabili intere (AA% BB%) e il nuovo rinvio alla subrouitine 380 evidenzia l'occupazione di (7/2=) 14 byte nell'area destinata alle variabili.

Riga 200

Nuovo incremento dei puntatori di fine variabili (6 dichiarate fino a questo momento).

Riga 220

Alterazione dei valori di due variabili precedentemente dichiarate in riga 170. Si noti che i puntatori rimangono inalterati dimostrando che, una volta che una variabile numerica viene dichiarata, successive modifiche del suo contenuto *non alterano* il numero di byte destinati all'occupazione da parte delle variabili interessate.

Righe 260-350

Utilizzando le stesse variabili che individuano i puntatori, e ricorrendo ad un ciclo For...Next con Step di 7 (multiplo di occupazione dei byte), vengono visualizzati i contenuti delle locazioni di memoria riservate alle variabili.

Ne approfittiamo per ricordare che il Chr$(14), nelle prime

righe del programma, serve per passare al set maiuscolo - minuscolo in modo da render più facilmente individuabili i nomi delle variabili; in caso contrario, infatti, verrebbero evidenziati caratteri semigrafici di difficile interpretazione.

La prima delle due lettere che compaiono (in reverse) rappresentano il nome della variabile. In parentesi sono raffigurati i loro valori del codice interno Commodore. L'ultimo dato (in reverse) del primo rigo di schermo rappresenta l'indirizzo del primo byte contenente il nome della variabile.

Al rigo successivo vengono visualizzati i contenuti dei sette byte interessati dalla variabile (indicata nel rigo di schermo precedente). Si noti con attenzione il modo di allocare il nome delle variabili. Si noti inoltre il fatto che, nel caso di variabili intere o stringa, alcuni byte (gli ultimi) sono *sempre* posti al valore nullo.

Anche in questo caso il lettore, data la versatilità del programma proposto, può divertirsi a modificare la riga 170 inserendo altre variabili intere, decimali, stringa dai nomi più bizzarri: il segmento di programma 370/470 visualizzerà in ciascun caso ciò che succede all'interno del calcolatore quando vengono dichiarate le tre tipologie di variabili.

Proviamo ora, avendo in memoria ancora il programma 2, a dichiarare le seguenti tre variabili dal nome piuttosto simile:

170 AA%=100:AA=100:AA$="100"

Una volta dato il consueto RUN, ci accorgiamo che, alla fine, nel primo caso il nome viene visualizzato con due lettere A maiuscole, nel secondo con due minuscole mentre nel terzo caso la prima è maiuscola e la seconda minuscola. Questo modo di alterare la prima, la seconda o entrambe le lettere delle variabili, consente all'interprete Basic di individuare senza equivoci le diverse variabili nel corso di una qualsiasi elaborazione.

Il terzo listato

```
100 REM  LISTATO N.3
110 :
120 REM SECONDA FASE: MODIFICA BYTE
130 REM RELATIVI A VARIABILI INTERE
```

```
140 :
150 AA%=100:PRINTCHR$(14)
160 PRINTCHR$(147)"PUNTATORI VARIABILE":GOSUB 38
    0
170 PRINT:PRINT:PRINT"1) MODIFICA AA%"
180 PRINT"2) MODIFICA PUNTATORI":PRINT"*) RITORN
    O AL MENU"
190 GET A$:IF A$="1" THEN GOSUB 240:GOTO 170
200 IF A$="2" THEN GOSUB 280:GOTO 170
210 GOTO 190
220 :
230 REM MODIFICA VALORE DI AA%
240 PRINT:INPUT "(*) AA%=";AA$:IF AA$="*" THEN R
    ETURN
250 AA=VAL(AA$):IF AA>32767 OR AA<-32768 THEN 24
    0
260 AA%=AA:GOSUB 380:GOTO 240
270 REM MODIFICA BYTE VAR. AA%
280 INPUT "(*) BYTE";AA$:AA=VAL(AA$)
290 IF AA$="*" THEN RETURN
300 IF AA>I+6 OR AA<I THEN 280
310 A1=AA
320 INPUT "(*) VALORE";AA$:AA=VAL(AA$)
330 IF AA$="*" THEN 280
340 IF AA>255 OR AA<0 THEN 320
350 POKE A1,AA:GOSUB 380:GOTO 280
360 :
370 REM VISUALIZZAZIONE BYTE VAR.INTERA AA%
380 I=PEEK(45)+PEEK(46)*256:X1=PEEK(I):X2=PEEK(I
    +1)
390 PRINTCHR$(18)I;X1;CHR$(X1)
400 PRINTCHR$(18)I+1;X2;CHR$(X2)
410 FOR J=I+2 TO I+6:PRINT"[RVS]"J"[RVOFF]"PEEK(
    J):NEXT
420 PRINT:PRINT"AA%="AA%;"   BB%="BB%;"   AB%="A
    B%;"   BA%="BA%
430 RETURN
440 END
```

Finora abbiamo assistito... passivamente al modo in cui il calcolatore gestisce la memoria RAM nel caso debba definire variabili di qualunque tipo. Vediamo ora che succede se noi, intenzionalmente, proviamo ad alterare il loro contenuto. A tale scopo digitiamo il programma N.3. E' ovvio che il lettore, con la propria fantasia, può apportare tutte le sofisticazioni che ritiene opportune.

Righe 160-210

Menu di scelta. Viene chiesta una delle due scelte possibili:

1) Alterazione del valore di AA% (dichiarata in riga 150)
2) Alterazione dei singoli byte costituenti la variabile.
Nel caso si scelga l'opzione 1, verrà richiesto un nuovo valore da attribuire alla variabile AA% (righe 240/260). E' ovvio che vengono rifiutati valori che escono dall'intervallo -32768 +32767 (riga 260). Subito dopo il valore digitato viene depositato in AA% e la subroutine 380/430 viene incaricata di visualizzare non solo il nome della prima variabile dichiarata (che, guarda caso, è proprio AA%), ma anche il contenuto dei singoli byte relativi ad essa, oltre al loro indirizzo.

Digitando il carattere di asterisco (*) si ritorna al menu principale. Se ora si sceglie l'opzione 2 (modifica puntatori) si avrà la possibilità di alterare il contenuto dei sette byte, uno alla volta. Modificandoli a caso, ma con criterio, si possono fare le seguenti deduzioni:
a/ Alterando il contenuto degli ultimi tre byte, il valore di AA% non viene in alcun modo modificato. Ciò dimostra che il Basic, nel caso di variabili intere, "guarda" esclusivamente il terzo ed il quarto byte.
b/ Alterando il contenuto di questi due byte si ottengono (riga 420) valori diversi di AA%
c/ Alterando i primi due byte (relativi cioè al nome della variabile) ci accorgiamo che il computer non riconosce più la prima variabile che incontra col nome AA%, ma col nome che le abbiamo attribuito! Grazie alla riga 420 vengono visualizzati i valori relativi a quattro variabili intere (AA% AB% BA% BB%): servono per verificare l'effettivo cambio di nome causato dalle Poke. Provate dunque (avendo davanti a voi la tabella del codice ASCII e addizionando 128 al valore del carattere tabellato) a modificare in tal senso i due byte relativi al nome, dapprima trasformando AA% in BB% (cioè 193-193 in 194-194) e poi in altre lettere qualunque.

```
100 REM  LISTATO N.4
110 :
120 REM  TERZA FASE : MODIFICA BYTE RELATIVI
130 REM  A VARIABILI IN VIRGOLA MOBILE
140 :
150 AA=100: PRINT CHR$ (14)
160 PRINT"[CLEAR]PUNTATORI VARIABILE": GOSUB  38
    0
170 PRINT"[DOWN]1) MODIFICA AA"
180 PRINT"2) MODIFICA PUNTATORI":PRINT
190 GET  A$: IF  A$="1"  THEN  GOSUB  240:GOTO
```

```
    170
200 IF  A$="2"  THEN  GOSUB  280: GOTO 170
210 GOTO 190
220 :
230 REM  MODIFICA VALORE DI AA
240 INPUT "[DOWN]AA=";AA$: IF  AA$="*"  THEN  RE
    TURN
250 AA=VAL(AA$)
260 GOSUB  380: GOTO  240
270 REM  MODIFICA BYTE VAR. AA
280 INPUT  "[DOWN]BYTE";AA$:X=VAL(AA$)
290 IF  AA$="*"  THEN  RETURN
300 IF  X >I+6  OR  X <I  THEN 280
310 A1=X
320 INPUT  "VALORE";AA$:X=VAL(AA$)
330 IF  AA$="*"  THEN  280
340 IF  X>255  OR  X<0  THEN 320
350 POKE  A1,X: GOSUB  380: GOTO  280
360 :
370 REM  VISUAL.BYTE VAR.VIRGOLA MOBILE AA
380 I=PEEK(45)+PEEK(46)*256: X1=PEEK(I): X2=PEEK
    (I+1)
390 PRINT "[RVS]" I;X1 CHR$ (X1)
400 PRINT "[RVS]" I+1 X2 CHR$ (X2)
410 FOR  J=I+2  TO  I+6: PRINT "[RVS]" J "[RVOFF
    ]" PEEK(J): NEXT
420 PRINT"AA="AA;"  BB="BB;"  AB="AB;"  BA="B
    A
430 RETURN
440 END
```

Il listato N.4 modifica i byte relativi a variabili decimali ed è sostanzialmente identico a quello N.3. Le differenze consistono nel trattamento di valori numerici diversi.

```
100 REM  LISTATO N.5
110 :
120 REM  QUARTA FASE: ESAME ALLOCAZIONE
130 REM  MATRICI VARIABILI INTERE E NON
140 PRINT CHR$(147)
150 X1=X1:X2=X2:X3=X3:X4=X4:X5=X5:I=I
160 :
170 DIM  AA%(12,34),CA(12),CX%(32)
180 :
190 PRINT CHR$(14)
200 X1=PEEK(47) + PEEK(48)*256: X2=PEEK(49)+PEEK
    (50)*256
210 PRINT "[RVS]ULTIMO BYTE OCCUPATO[RVOFF]" X2
220 PRINT"[RVS]"X1 PEEK(X1) CHR$(PEEK(X1)) "[RVO
    FF] NOME"
```

```
230 PRINT "[RVS]" X1+1 PEEK(X1+1) CHR$(PEEK(X1+1
    )) "[RVOFF] VETTORE"
240 PRINT "[RVS]" X1+2 PEEK(X1+2) "[RVOFF] BYTE
    +"
250 PRINT "[RVS]" X1+3 PEEK(X1+3) "[RVOFF] *256
    OCCUPATI =";
260 X4=PEEK(X1+2)+PEEK(X1+3)*256:PRINTX4
270 X5=X1+PEEK(X1+2)+PEEK(X1+3)*256
280 PRINT "(OCCUPAZIONE FINO A" X5-1 ")"
290 PRINT "[RVS]" X1+4 "[RVOFF]" PEEK(X1+4)" DIM
    ENSIONI=";
300 X3=X1+4: PRINT "[RVS]" PEEK(X3)
310 FOR  I=1  TO  PEEK(X3)*2: PRINT"[RVS]"X3+I+1
    ;PEEK(X3+I):NEXT
320 IF  X2-X1 = X4  THEN  370
330 X1=X5: REM  A. DE SIMONE DIDATTICA '84
340 PRINT "[RVS]PREMI UN TASTO"
350 IF  PEEK(197)=64  THEN  350
360 GOTO  220
370 LIST  170
```

Il listato N.5, che non viene descritto in maniera approfondita, esamina l'allocazione dei vettori e segue le regole prima descritte. Il lettore può verificarlo alterando la riga 170 inserendo più vettori o dimensionandoli diversamente. L'automatismo del programma consente di esaminare, byte dopo byte, ciascun valore dichiarato. Diremo soltanto che, anche in questo caso, i primi vettori che si incontrano sono proprio quelli dichiarati seguendo l'ordine "logico" del programma.

```
100 REM  LISTATO N.6
110 :
120 REM  STUDIO ALLOCAZIONE STRINGHE
130 :
140 AA$ = "STRINGA"
150 X1=X1: X2=X2: I=I: PRINT "[CLEAR]"
160 X1 = PEEK(45)+PEEK(46)*256
170 X2 = PEEK(47)+PEEK(48)*256
180 PRINT X1 "[RVS]" CHR$(PEEK(X1)) "[RVOFF] NOM
    E"
190 PRINT X1+1 "[RVS]"CHR$(PEEK(X1+1)) "[RVOFF]
    STRINGA"
200 PRINT X1+2 "LUNG. STRINGA =[RVS]" PEEK(X1+2)
210 PRINT X1+3 "LA.INDIR.STRINGA=[RVS]" PEEK(X1+
    3)
220 PRINT X1+4 "HA.INDIR.STRINGA=[RVS]" PEEK(X1+
    4)
```

```
230 PRINT"[DOWN][RVS]("  PEEK(X1+3) "[LEFT] +" PE
    EK(X1+4)"[LEFT] * 256 =";
240 X2=PEEK(X1+3)+PEEK(X1+4)*256:PRINTX2"[LEFT])
    [DOWN]"
250 FOR  I=0  TO  PEEK(X1+2)-1
260 PRINT "[RVS]"X2+I"[RVOFF]" CHR$(PEEK(X2+I)):
     NEXT
270 LIST  140
```

In quest'ultimo listato (N.6) viene esaminato il luogo in cui i caratteri delle singole stringhe dichiarate vengono depositati. Il lettore può provare ad alterare il contenuto della variabile AA$ (riga 140) oppure a modificarlo ricorrendo a concatenazioni di più stringhe. Uno studio particolareggiato sull'allocazione dei vettori stringa non viene indicato dato che, a questo punto, il lettore, seguendo la falsariga dei programmi pubblicati in queste pagine, può farlo da solo.

Un breve riepilogo

- Nel C/64 la prima locazione di memoria a disposizione è la 2049; nel seguito dell'inserto verrà indicata con LI (Locazione Inizio).
- L'ultima locazione occupata da un programma Basic varia, come è intuitivo, a seconda della lunghezza del programma stesso, e verrà indicata con LF (Locazione Fine).
- Il cosiddetto "indirizzo" di partenza del programma Basic viene calcolato esaminando il contenuto di due locazioni di memoria (puntatori): la 43 e la 44.
- Per conoscere, dunque, l'indirizzo di partenza del programma presente in memoria, sarà sufficiente eseguire il calcolo:

*LI=PEEK(43)+PEEK(44)*256*

In genere il risultato porterà ai valori prima visti (2049 nel C/64).

- Altre due locazioni (la 45 e la 46) contengono, in "codice", l'indirizzo dell'ultima locazione interessata dal programma Basic presente in memoria in quel momento:

*LF=PEEK(45)+PEEK(46)*256*

LF, come già detto, cambia a seconda della lunghezza del programma presente.

• Non appena si accende l'apparecchio LI coincide con LF.
• Col termine "Puntatore" si intende il valore numerico della locazione interessata in un'operazione. Ad esempio le locazioni 43 e 44 "puntano" all'inizio del programma Basic mentre le 45 e 46 puntano alla sua fine.

L'esigenza dell'overlay

A volte capita, anche se è molto raro, che alcuni programmi non possono essere ospitati in un computer perchè sono più lunghi della memoria disponibile. Se infatti si cerca di digitarli una linea alla volta, queste vengono accettate fino a che è possibile: tentando di proseguire si ottiene null'altro che un messaggio di "Out of memory error" che impedisce il proseguimento della digitazione.
Analogamente programmi anche molto brevi, e di conseguenza caricabili o digitabili senza difficoltà alcuna, contengono alcune istruzioni del tipo DIM che, per essere eseguite, richiedono una certa quantità di memoria.
Sembrerebbe che, nei casi appena visti, sia impossibile utilizzare i programmi stessi avendo a disposizione una quantità modesta di RAM.
Molto spesso, però, un certo numero di linee di programma vengono usate per "inizializzare" il programma stesso e non occorrono più nel resto dell'elaborazione. Si può, allora, caricare tale parte di programma, farla girare, ed in seguito caricare ed utilizzare la seconda parte.
Se però si agisce come è stato descritto, al momento di impartire il secondo RUN (in seguito al caricamento della seconda parte), si azzerano tutti i valori delle variabili inizializzate precedentemente.
Per non perdere il contenuto delle variabili, la Commodore specifica che se si inserisce, all'interno di un listato, una riga del tipo...

XXX Load"seconda parte",8

...il programma con tale nome viene caricato e "lanciato" senza alterare in nessun modo le variabili elaborate dalla prima parte del programma stesso. Possiamo iniziare i nostri esperi-

menti digitando i seguenti due programmi che differiscono tra loro in minima parte:

```
100 print"questo è"
110 print"il programma"
120 print"alfa"
130 geta$:ifa$=""then130
140 ifa$="b"then160
150 print:goto100
160 load"beta",8
170 print 2*3-67
180 print "frase"
```

Il listato qui sopra riportato deve essere memorizzato su disco con il nome "alfa". Dopo averlo registrato, sostituite (righe 120 e 160) il nome "Alfa" con "Beta", e viceversa, e registratelo nuovamente con il nome "beta":

```
100 print"questo è"
110 print"il programma"
120 print"beta"
130 geta$:ifa$=""then130
140 ifa$="a"then160
150 print:goto100
160 load"alfa",8
170 print 2*3-67
180 print "frase"
```

Si noti che la lunghezza dei due programmi è rigorosamente identica; ve ne potete accorgere richiedendo Print Fre(0) che deve fornire due valori assolutamente eguali.

Se, ora, fate girare uno qualsiasi dei due programmi, noterete due cose molto importanti:

• premendo un tasto qualsiasi, apparirà il messaggio che indica quale dei due programmi è presente in memoria; premendo "A" (oppure "B") verrà invece caricato l'altro listato e immediatamente fatto partire.

• le istruzioni presenti nelle righe 170 e 180 (successive, in altre parole, alla riga contenente Load) non vengono mai eseguite: non appena un programma è caricato, questo parte dalla SUA prima riga.

A parte questo, non si verificano inconvenienti di sorta proprio perchè i due listati, volutamente banali, occupano la stessa area di memoria. Provate, però, ad effettuare il seguente esperimento:

- spegnete e accendete il computer
- digitate il seguente listato:

```
100 rem differenze
110 :
120 print:print"1- eseguo differenze"
130 print"2- carico somme"
140 geta$:ifa$=""then140
150 ifa$="1"thengosub180
160 ifa$="2"then240
170 print:goto120
180 print:print"(0= ritorna)"
190 input"minuendo";x
200 ifx=0thenprint:return
210 input"sottraendo";y
220 print"differenza="x-y
230 print:goto180
240 load"somme",8
```

- registratelo con il nome "Differenze".
- cancellatelo, digitate il seguente e registratelo con il nome "somme":

```
100 rem somme
110 :
120 print:print"1- eseguo somme"
130 print"2- carico differenze"
140 geta$:ifa$=""then140
150 ifa$="1"thengosub180
160 ifa$="2"then240
170 print:goto120
180 print:print"(0= ritorna)"
190 input"primo addendo";x
200 ifx=0thenprint:return
210 input"secon.addendo";y
220 print"somma="x+y
230 print:goto180
```

```
240 load"differenze",8
250 remquesto programma, come si può notare, è più lungo del
listato
260 remchiamato "differenze" a causa, se non altro, di queste
righe rem
```

• caricate il programma "differenze" e date il Run
• fate in modo che, premendo il tasto "2", venga caricato il programma "somme".
• premete il tasto Run/Stop e chiedete il List... Sorpresa! Che fine hanno fatto le altre righe Rem e che cosa rappresenta quella riga "strana" prima della fine?

Ma le sorprese non sono finite... Provate a digitare un semplice:

300 rem

...lo schermo si sconvolge e non è possibile recuperare il programma nemmeno con i tasti Run/Stop e Restore. E' necessario spegnere e riaccendere il computer.

Il richiamo di altri programmi, durante l'elaborazione di un programma, è possibile ma è indispensabile che il programma richiamato sia più breve (o al massimo eguale) di quello che lo "chiama"; vedremo tra breve il perchè.

Allo scopo di capire in quali casi la procedura può essere applicata e in quali, al contrario, può dar luogo ad inconvenienti di vario tipo, consigliamo di seguire *alla lettera* le fasi qui di seguito indicate.

Al termine della lettura sarete sorpresi della semplicità con cui è possibile aumentare le potenzialità di un personal computer Commodore.

Primo esperimento

Fase 1) Accendete il computer oppure, se è acceso, spegnetelo e riaccendetelo in modo da esser sicuri della sua corretta inizializzazione.

```
100 rem prova n.1/a
110 :
120 print chr$(18)"inizio prova 1/a"
130 a=1:b=2:c=3:d=4
140 e=1.2: f=2.3
150 g=3.4: h=5.6
```

```
160 print a;b;c;d;
170 print e;f;g;h
180 print chr$(18)"fine prova 1/a"
190 :
200 rem queste tre righe
210 rem servono solo per
220 rem allungare il brodo
230 :
240 load "prova n.1/b",8
```

Fase 2) Digitate il programma n.7 (dal nome "Prova N.1/a") così come è pubblicato, tranne la riga 240. Controllate che funzioni correttamente, digitate la riga 240 e registratelo su di un disco oppure su di un nastro (che numerate con 1). E' ovvio che la sintassi riportata è valida per chi utilizza l'unità a dischi. Chi si serve del registratore a cassette trascriverà semplicemente:

```
240 Load"Prova n.1/b"
```

Tale avvertenza (eliminazione di ",8") vale per tutti i brevi listati presentati nel caso si desideri ricorrere al registratore.

Fase 3) Spegnete e riaccendete il computer. Digitate il programma n.8, provatelo (ottenendo una serie di zeri), e registratelo col nome "Prova n.1/b" nel modo consueto sullo stesso disco oppure su un altro nastro (N.2): Utilizzando due nastri si semplificheranno le varie operazioni che stiamo per descrivere.

```
100 rem prova n.1/b
110 print
120 print "inizio prova 1/b"
130 print a;b;c;d;
140 print e;f;g;h
150 print "fine prova 1/b"
```

Fase 4) Caricate, da nastro o disco, il programma "Prova n.1/a" e digitate RUN. Ciò che accade è piuttosto semplice: Dapprima verrà elaborato il programma presente in memoria; in seguito (riga 240) verrà caricato il listato "Prova n.1/b". La successiva visualizzazione delle stesse variabili di prima dimostra che, nonostante sia stato caricato un nuovo programma, le va-

riabili dichiarate precedentemente non vengono in alcun modo alterate. Se infatti, all'apparizione del consueto Ready chiediamo il listato, appare null'altro che quello n.8, vale a dire quello caricato e lanciato grazie alla riga 240. Sembrerebbe che la Commodore abbia proprio ragione. Ma... è davvero tutto in ordine?

Fase 5) A questo punto, (vale a dire avendo in memoria il programma "Prova n.1/b" caricato dal programma "Prova n.1/a"), aggiungiamo una linea qualunque come la:

160 rem commodore comput.club

Se ora chiediamo il listato, esso apparirà più lungo di una sola riga e comunque più breve di quello n.7.

Fase 6) Registriamo il programma così formato, con lo stesso nome di prima in modo che possa esser richiamato nuovamente dal programma "Prova N.1/a".

Il modo di effettuare questa operazione dovrebbe esser noto ma lo riassumiamo brevemente:

Open 15,8,15,"S:Prova N.1/a":close1
Save"Prova n.1/b",8

nel caso si possegga un drive 1541, e:

Save"Prova n.1/b"

utilizzando il nastro 2 (vale a dire cancellando il precedente programma ivi registrato) nel caso si usi il registratore.

Fase 7) Spegnete, riaccendete e caricate da nastro o disco il "vecchio" programma (Prova n.1/a).

Impartite il RUN. Le istruzioni del primo programma, come è facile verificare, vengono correttamente eseguite. Non appena, però, viene caricato e lanciato il secondo programma, ci accorgiamo che il risultato è diverso da quello visto nella fase 4. Ciò è accaduto, come vedremo tra breve, perchè il secondo programma è più lungo del primo e provoca gli stessi inconvenienti dei due programmi "Somme" e "Differenze".

Come è possibile che ciò sia avvenuto? Chiediamo il listato (List): essò è proprio identico a quello che ci aspettiamo con in

più la sola riga aggiunta nella fase 5: è comunque ben più breve del primo!

Ebbene il secondo programma *sembra* più breve del primo ma risulta *effettivamente* più lungo. Vediamo di spiegarcene il perchè.

Torniamo ad esaminare la fase 4 e confrontiamola con quanto segue: quando il programma "Prova n.1/a", giunto alla riga 240, carica il programma "Prova n.1/b", questo viene allocato a partire da LI. I puntatori di LF rimangono però *inalterati*. Ciò è dovuto al fatto che le variabili precedentemente dichiarate iniziano a partire da LF.

Se un programma, ad un certo punto di un'elaborazione, "chiama" un altro programma, questo, dopo il caricamento, viene considerato lungo quanto il precedente anche se è composto da una sola riga! Pertanto l'interprete Basic "vede", dopo il caricamento di un programma, un listato che termina in LF e se, a questo punto, aggiungiamo una qualsiasi riga, questa viene allocata come di consueto, ed i puntatori di fine Basic, di conseguenza, alterati come se fosse stata aggiunta ad un programma più lungo.

Quanto descritto è proprio l'errore che (intenzionalmente) vi abbiamo fatto commettere aggiungendo la riga 160 nella fase 5. Questa è stata infatti aggiunta *dopo* che il programma "lungo" era stato caricato automaticamente da "Prova n.1/a". Ciò dimostra che, effettivamente, nel caso di ricorso a tecniche di overlay, è indispensabile che il primo programma da caricare in memoria sia il più lungo di quello (o quelli) che in segito saranno richiamati in cascata e, soprattutto, che non vengano modificati i programmi durante tali delicatissime fasi.

La spiegazione dell'inconveniente è piuttosto semplice da comprendere: dopo l'area Basic inizia l'area delle variabili. Se "invadiamo" una zona di quest'ultima con un programma, la "coda" di questo ne cancella alcune ed esattamente quelle che per prime erano state dichiarate nel corso dell'elaborazione del primo programma.

Verifica del primo esperimento

1) Per verificare quanto è stato affermato, digitare il programma "Prova n.2/a" privo della linea 280; dopo aver verificato che funziona correttamente, digitare la riga 280 e registrarlo col nome "Prova n.2/a".

```
100 rem prova n.2/a
110 :
120 print "inizio prova 2/a"
130 a=1:b=2:c=3:d=4
140 e=1.2: f=2.3
150 g=3.4: h=5.6
160 print a;b;c;d;
170 print e;f;g;h
180 print "inizio memoria =";
190 print peek(43)+peek(44)*256
200 print "fine memoria =";
210 print peek(45)+peek(46)*256
220 print "fine prova n.2/a"
230 :
240 rem queste tre righe
250 rem servono solo per
260 rem allungare il listato
270 :
280 load "prova n.2/b",8
```

2) per sicurezza spegnete, riaccendete e digitate il programma "Prova n.2/b".

```
100 rem prova n.2/b
110 :
120 print "inizio prova 2/b"
130 print a;b;c;d;
140 print e;f;g;h
150 print peek(43)+peek(44)*256
160 print peek(45)+peek(46)*256
170 print "fine prova 2/b"
```

Una volta controllatone il corretto funzionamento, trascriviamo su di un foglio di carta i valori forniti dalle righe 150 e 160. Registratelo dunque col nome "Prova n.2/b" in modo che in seguito possa esser richiamato da programma.

3) Carichiamo "Prova n.2/a" e "lanciamolo". Al termine dell'elaborazione (dopo cioè che viene caricato e lanciato "Prova n.2/b", riga 280), ci accorgiamo che il secondo programma, benchè più breve del primo, possiede gli stessi puntatori di fine Basic. In ogni caso i puntatori indicano valori maggiori di quelli precedentemente trascritti sul foglio di carta.

4) Col programma "Prova n.2/b" (che è ora allocato in memoria perchè richiamato da "Prova n.2/a"), aggiungiamo la riga:

180 Rem Commodore Computer Club
oppure una qualsiasi altra riga.
5) Digitiamo RUN e Return. I valori delle variabili (A, B, C, D, E, F) sono ovviamente tutti nulli non solo perchè è stato dato il RUN, ma anche perchè è stata aggiunta una riga. Ricordiamo, per inciso, che l'inserimento, la cancellazione o la semplice modifica di una riga del programma azzera sempre e comunque qualsiasi variabile memorizzata.

Possiamo dunque notare che i puntatori di fine Basic (elaborazione di riga 160) vengono incrementati.

Conclusioni sul primo esperimento

• In un programma è possibile, mantenendo inalterate le varibili numeriche, dare l'ordine di caricare un altro programma, purchè quest'ultimo sia più breve o al massimo di eguale lunghezza di quello "chiamante".
• Il programma chiamato non deve in nessun caso essere manipolato, pena la perdita della caratteristica di "brevità".
• Nel caso si desideri modificare il programma chiamato, procedere come segue:
- Spegnere e riaccendere il computer.
- Caricare il programma che si desidera modificare.
- Modificarlo e registrarlo nuovamente con lo stesso nome di prima avendo l'accortezza di verificare che le modifiche apportate non l'abbiano reso più lungo del programma che, in seguito, lo chiamerà.

• Per conoscere la quantità di byte occupata da un programma, digitare *non il semplice* Print Fre(0) ma:
Clr: Restore: Print Fre(0) (Return).

Se il valore che risulta è negativo, digitate:

Print 65536 + Fre(0)

La gestione delle variabili stringa

I computer Commodore allocano le variabili stringa, a seconda dei casi, in due zone della memoria. Una di queste si trova *al di là* dell'indirizzo di fine Basic. Alcune variabili, invece, vengono individuate, dall'interprete, all'interno del programma Basic stesso.

Tale metodo di memorizzazione dei dati (tipico delle sole variabili stringa), che tra breve verrà in parte descritto, ha lo scopo di ottimizzare l'occupazione della memoria RAM. Vediamo come:

Se in un programma Basic figura una linea del tipo:

100 a$="abcde": print a$

l'interprete non ha motivo di depositare i caratteri alfanumerici "abcde" in una zona di memoria "esterna" a quella occupata dal programma Basic. Poichè, infatti, tali caratteri si trovano di già in una zona RAM, a quale scopo depositarli da qualche altra parte, creando un inutile doppione? In altre parole quando il computer incontra un'istruzione del tipo Print A$, andrà a rintracciare l'area della memoria alla quale sono associati i vari caratteri. Nel caso del microprogramma appena visto, questa si trova esattamente ad otto byte di distanza dall'inizio del programma stesso, vale a dire (ma dovreste ormai saperlo): 2 byte di link, 2 di numerazione Basic, 2 del nome della variabile, 1 per il simbolo (=) ed uno, infine, per il simbolo degli apici (").

Ben diverso è il caso della linea Basic:

100 a$="abc"+"def": print a$

In questo caso (come in tutti i casi di manipolazione di stringhe) il computer *deve* allocare la stringa A$ (ottenuta come elaborazione di due gruppi di caratteri) in qualche parte della memoria, *diversa* da quella relativa ai programmi Basic. Una cosa è infatti disporre dell' indirizzo di partenza in cui si trovano *tutti in fila* i caratteri relativi alla stringa, altra cosa è invece disporre dell'indirizzo in cui è situata un'operazione di concatenazione di stringhe. Per sincerarcene, digitate e fate girare i due micro-programmi seguenti:

100 print fre(0)
110 a$="abcdefghij": print a$: print fre(0)

... e questo...

100 print fre(0)
110 a$="abcde"+"fghij": print a$: print fre(0)

Quest'ultimo sembra essere più lungo del precedente di appena tre byte, cioè i caratteri apici, più, apici ("+"). Infatti è proprio così e ce lo dimostra il valore di FRE(0) che appare pri-

ma della visualizzazione di "abcdefghij". Il secondo valore che appare, indica, invece, il numero di byte a disposizione *dopo* che la variabile A$ è stata dichiarata.

Mentre nel primo programma la variabile A$ è all'interno del programma stesso, nel caso successivo A$ costituisce il risultato di una manipolazione e viene pertanto situata automaticamente in altra zona occupando, di conseguenza una quantità di memoria maggiore, rispetto al precedente microprogramma, del numero di caratteri costituenti la stringa somma.

Conclusioni

1) Se in un programma viene definita una variabile stringa, essa non occupa altra zona se non quella all'interno del programma Basic stesso e viene individuata da puntatori che, nonostante siano situati, come tutti i puntatori, *al di fuori* dell'area destinata al Basic puntano a quelle locazioni di memoria.
2) Se la variabile stringa è il frutto, invece, di una manipolazione (Right$ Left$ ecc.) oppure di una somma (A$+B$), essa viene allocata, come i suoi puntatori, in una zona situata al di fuori dell'area riservata ai programmi Basic.

Una descrizione dettagliata di come il Basic gestisce le stringhe ed i suoi puntatori esula dallo scopo del presente inserto.

(Continuazione e fine al prossimo numero)